NASA/TM-2000-208641/VER1.0/VOL#

ICESat (GLAS) Science Processing Software Document Series

Volume
GSAS Detailed Design Document
Version 1.0

Jeffrey Lee/Raytheon ITSS
Observational Science Branch
Laboratory for Hydrospheric Processes
NASA/GSFC Wallops Flight Facility
Wallops Island, Virginia 23337

November 2000

ICESat Contacts:

H. Jay Zwally, ICESat Project Scientist
NASA Goddard Space Flight Center
Greenbelt, Maryland 20771

Bob E. Schutz, GLAS Science Team Leader
University of Texas Center for Space Research
Austin, Texas 78759-5321

David W. Hancock III, Science Software Development Leader
NASA/GSFC Wallops Flight Facility
Wallops Island, Virginia 23337

Foreword

This document describes the detailed design of GLAS Science Algorithm Software.

The GEOSCIENCE LASER ALTIMETER SYSTEM (GLAS) is a part of the EOS program. This laser altimetry mission will be carried on the spacecraft designated EOS ICESat (Ice, Cloud and Land Elevation Satellite). The GLAS laser is a frequency-doubled, cavity-pumped, solid state Nd:YAG laser.

This document was prepared by the Observational Science Branch at NASA GSFC / WFF, Wallops Island, VA, in support of B. E. Schutz, GLAS Science Team Leader for the GLAS Investigation. This work was performed under the direction of David W. Hancock, III, who may be contacted at (757) 824-1238, hancock@osb1.wff.nasa.gov (e-mail), or (757) 824-1036 (FAX).

The following GLAS Team members contributed to the creation of this document:

RITSS / Kristine Barbieri

RITSS / Suneel Bhardwaj

RITSS / Anita Brenner

972 / David W. Hancock, III

RITSS / Peggy Jester

RITSS / Jeff Lee

RITSS / Gladstone Marcus

RITSS / Carol Purdy

RITSS / Lee Anne Roberts

Table of Contents

List of Figures

List of Tables

Section 1

Introduction

1.1 Identification of Document

This document is identified as the GLAS Science Algorithm Software (GSAS) Detailed Design Document. The unique document identification number within the GLAS Ground Data System numbering scheme is *TBD*. Successive editions of this document will be uniquely identified by the cover and page date marks.

1.2 Scope of Document

The GLAS I-SIPS Data Processing System, show in Figure 1-1, provides data processing and mission support for the Geoscience Laser Altimeter System (GLAS). I-SIPS is composed of two major software components - the GLAS Science Algorithm Software (GSAS) and the Scheduling and Data Management System (SDMS). GSAS processes raw satellite data and creates EOS Level 1A/B and 2 data products. SDMS provides for scheduling of processing and the ingest, staging, archiving and cataloging of associated data files. This document describes the detailed design of GSAS and contains information specific to the Version 1 delivery of the software.

1.3 Purpose and Objectives of Document

This document describes the detailed design of the GLAS Science Algorithm Software. It contains descriptions, data flow diagrams, structure charts, and state transition diagrams for each major component of the GSAS.

The purpose of this document is to present the detailed design of the GSAS. It is intended as a reference source which would assist the maintenance programmer in making changes which fix or enhance the documented software.

1.4 Document Status and Schedule

The GLAS Science Algorithm Software Detailed Design Document is currently released as Version 1.0.

1.5 Document Organization

This document's outline is assembled in a form similar to those presented in the NASA Software Engineering Program [Information Document 2.3a].

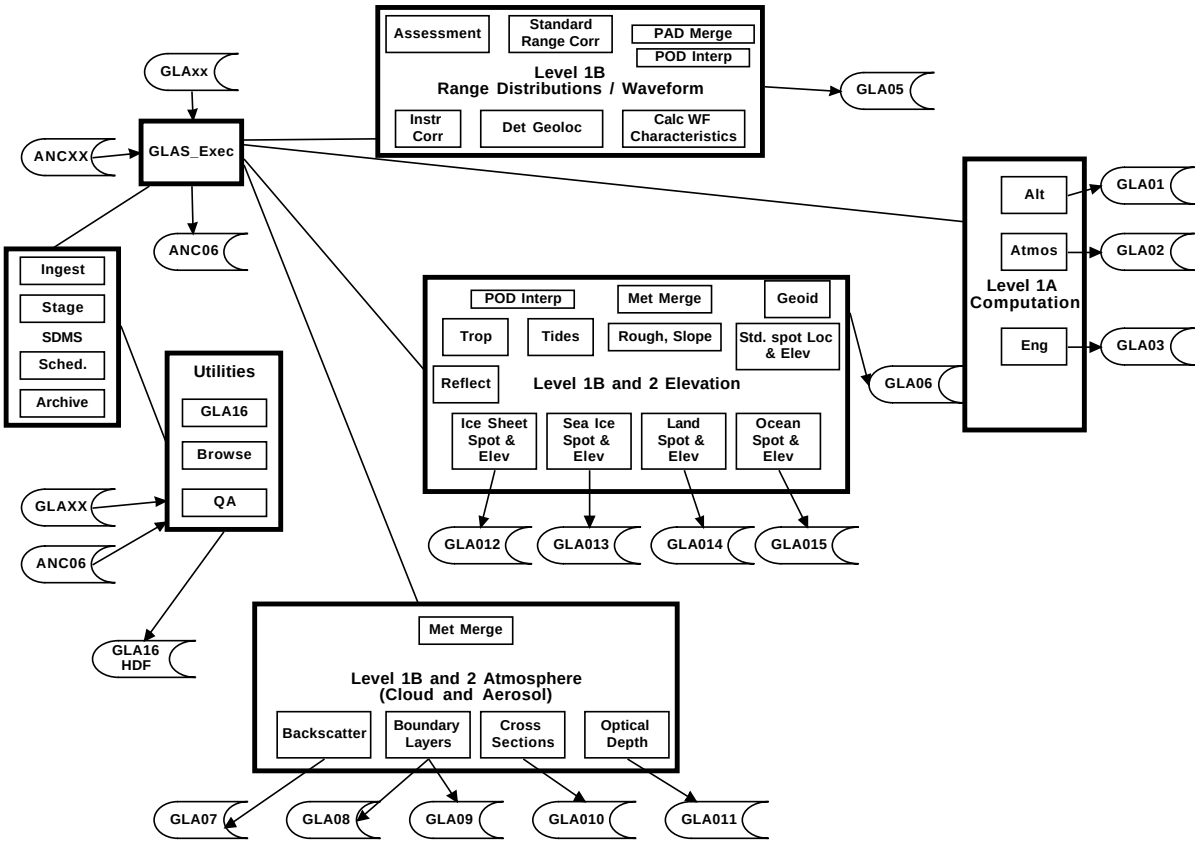

Figure 1-1 I-SIPS Software Top-Level Decomposition

1.6 Document Change History

Document Name: GLAS Science Algorithm Software Detailed Design Document		
Version Number	**Date**	**Nature of Change**
Version 0	August 1999	Original Version
Version 1	November 2000	Revised for V1 software.

Section 2

Related Documentation

2.1 Parent Documents

Parent documents are those external, higher-level documents that contribute information to the scope and content of this document. The following GLAS documents are parent to this document.

a) *GLAS Science Software Management Plan* (GLAS SSMP), Version 3.0, August 1998, NASA Goddard Space Flight Center, NASA / TM-1999-208641 / VER3 / VOL1.

b) *GLAS Science Data Management Plan* (GLAS SDMP), Version 4.0 July 1999, NASA Goddard Space Flight Center, NASA / TM-1999-208641 / VER4 / VOL2.

c) *GLAS Science Software Requirements Document* (GLAS SSRD), Version 2.1 August 2000, NASA Goddard Space Flight Center.

d) *GLAS I-SIPS Software Architectural Design Document*, Version 2.0, NASA Goddard Space Flight Center, October 1998.

2.2 Applicable Documents

Applicable documents include reference documents that are not parent documents. This category includes reference documents that have direct applicability to, or contain policies binding upon, or information directing or dictating the content of this document. The following GLAS, EOS Project, NASA, or other Agency documents are cited as applicable to this architectural design document.

a) *Data Production Software and Science Computing Facility (SCF) Standards and Guidelines*, January 14, 1994, Goddard Space Flight Center, 423-16-01.

b) *EOS Output Data Products, Processes, and Input Requirements*, Version 3.2, November 1995, Science Processing Support Office.

c) *NASA Earth Observing System Geoscience Laser Altimeter System GLAS Science Requirements Document*, Version 2.01, October 1997, Center for Space Research, University of Texas at Austin.

d) *Precision Orbit Determination (POD),* Algorithm Theoretical Basis Document, Version 0.1, December 1996, Center for Space Research, The University of Texas at Austin.

e) *Atmospheric Delay Correction to GLAS Laser Altimeter Ranges,* Algorithm Theoretical Basis Document, Version 0.3, December 1996, Massachusetts Institute of Technology.

f) *Algorithm Theoretical Basis Document for the GLAS Atmospheric Channel Observations*, Version 0 (Preliminary), December 1995, Goddard Space Flight Center.

g) *Geoscience Laser Altimeter System: Surface Roughness of Ice Sheets,* Algorithm Theoretical Basis Document, Version 0.3, December 1996, University of Wisconsin.

h) *Determination of Sea Ice Surface Roughness from Laser Altimeter Waveform,* Algorithm Theoretical Basis Document, Version 0 (Preliminary), December 1995, The Ohio State University.

i) *Laser Footprint Location and Surface Profiles*, Algorithm Theoretical Basis Document, Version 0 (Preliminary), December 1996, Center for Space Research, The University of Texas at Austin.

j) *Precision Attitude Determination (PAD)*, Algorithm Theoretical Basis Document, December 1996, Center for Space Research, The University of Texas at Austin.

2.3 Information Documents

The following documents are provided as sources of information that provide background or supplemental information that may clarify or amplify material presented in this document.

a) *NASA Software Documentation Standard Software Engineering Program*, NASA, NASA-STD-21000-91, July 29, 1991.

b) *Science User's Guide and Operations Procedure Handbook for the ECS Project, Volume 4: Software Developer's Guide to Preparation, Delivery, Integration and Test with ECS*, Final, August 1995, Hughes Information Technology Corporation, 205-CD-002-002.

c) *GSAS Users Guide, Version 1*, NASA Goddard Space Flight Center, September 2000.

d) *GLAS Level 0 Instrument Data Product Specification*, Version 2.2, March 17, 1998, NASA Goddard Space Flight Center Wallops Flight Facility, GLAS-DPS-2610.

e) *GLAS Standard Data Products Specification - Level 1*, Version 2.0, January 1999, NASA Goddard Space Flight Center Wallops Flight Facility, GLAS-DPS-2621.

f) *GLAS Standard Data Products Specification - Level 2*, Version 2.0, March 17, 1998, NASA Goddard Space Flight Center Wallops Flight Facility, GLAS-DPS-2641.

g) *Data Production Software, Data Management, and Flight Operations Working Agreement for AIRS, AMSU-A and MHS/AMSU-B*, NASA Goddard Space Flight Center, January 1994.

Section 3

Design Issues

3.1 Requirements

GSAS was designed with many specific and several generic requirements in mind. These requirements may be found in the GLAS Software Requirement Document. Several of the more critical requirements are listed here:

- The software will be designed for maximum portability and code-reuse.
- All standard data products will produced in an integer-binary format.
- Input and output products will be delimited by start and stop times.
- Full processing history will be available via metadata.
- Standardized messaging and error-handling using local ancillary files will be available to all subprocesses.
- Changeable parameters will be defined in local ancillary files.
- Capability to fully and partially process and reprocess data with several different scenarios, including:
  - One processing string to that starts with GLAS telemetry data (GLA00) as input to create all output L1A products (GLA01-04).
  - One processing string to that starts with L1A altimetry data (GLA01) as input to create an output waveform product (GLA05).
  - One processing string that starts with a waveform product (GLA05) input as to produce output elevation products (GLA06, 12,13,14,15).
  - One processing string that starts with L1A atmosphere (GLA02) input and produces output atmosphere products (GLA07,08,09,10,11).
  - One processing string that starts with a waveform product (GLA05) as input to produce a primary elevation product (GLA06).

3.2 Single vs. Multiple Executables

During the design phase, both single-program and multiple-program approaches were considered. Advantages of the single program approach include a potentially reduced I/O burden (dependent on the operational use of the software), less software maintenance, and ease of configuration management. The disadvantages of this approach are slightly more code complexity and a potential for increased memory usage due to non-active global memory. The advantages of multiple programs are simplicity and relatively reduced memory usage. The disadvantages are an increased software maintenance burden and increased configuration management difficulty.

By keeping the top-level code generic and developing on the single-program approach, the Team retains maximum development flexibility. If there is a solid reason not to implement the GSAS processing system as a single program, it will be very easy to subset the software and create four independent programs. This will allow the Team to take advantage of new information as development progresses into prototyping and load testing with a minimum of wasted effort.

3.3 Software Reuse

Even using a single-executable approach for the main processing software, the team recognizes that there will be other task–specific software which will interface with data created by the I-SIPS data processing system. In order to effect the reuse of this software, the GLAS Team will implement major components and subsystems as shared libraries. These libraries are generic so they may be used without modification. It is intended that associated utilities will be written to use these libraries in order to make full benefit of code-reuse.

3.4 I/O and Unit Conversion

The software reuse approach was especially important in the design of the GLAS Product input/output routines. The I/O routines were designed in a modular fashion to make them available for use in software outside of GLAS_Exec. All input/output statements are implemented in product-specific subroutines. All data transformation (scaling from integer to floating point and vice versa) are implemented in product-specific routines. This insures consistency in the conversion process methodology and forces a great deal of granularity in the design. Additionally, care was taken to minimize the number of support routines required by the I/O conversion processes in order to maximize the potential for software reuse.

3.5 Invalid Values

Not all data received from GLAS will be suitable for science processing. In addition, given the nature of the raw telemetry packets, some data may be missing. The concept of an “invalid value” is used to signify that data is invalid or missing and should not be used for processing. Invalid values are datatype-specific values which are defined in the GLAS global constants module. These variables are assigned to Product File variables in order to indicate invalid or missing data. The current values are defined in Table 3-1.

Table 3-1 Invalid Values

Datatype	Invalid Value
1 byte integer	127
2 byte integer	32767
4 byte integer	2147483647

Table 3-1 Invalid Values

Datatype	Invalid Value
4 byte real	x7F7FFFFF
8 byte real	x7FEFFFFFFFFFFFF

3.6 Reprocessing and Pass-Thrus

Reprocessing and partial-processing requirements dictated great care in the design of GSAS. In addition to executing all science algorithms consecutively, it is required that GSAS be able to run selective science algorithms with varying input data. Processing with a selected set of science algorithms and products is defined as a specific processing “scenario”. Several sample scenarios are listed in the requirements section above. The software not only must execute specific science algorithms, it is required to rewrite specific products, partially replacing selected data. An example of this is replacing the orbit on the primary elevation product (GLA06).

In order to accommodate the reprocessing requirement, GLAS_Exec is designed to use “pass-thru” data management. The “pass-thru” concept dictates that common data are passed from lower-numbered products to higher-numbered products on input. In the design, the product data structures can be input, output or both. Science algorithms are required to use input data from the highest-numbered product possible and pass computed data to requisite higher-numbered products.

3.7 Data Buffering

Data buffering is a fairly complex process. GSAS is required to process data one second at a time without buffering, except in one case. The Atmosphere subsystem ATBD has required that data be buffered to twenty seconds. This buffering has been designed into the Atmosphere subsystem, such that other portions of the software are not impacted by the added complexity. However, during the implementation it was decided to minimize the buffering complexity by adopting a constraint such that GLA08-11 will not be processed independently of one another. This constraint somewhat limits the granularity of re-processing, but was approved by the GLAS Change Control Board as an acceptable trade-off. The buffering concept is fully documented in the Atmosphere section.

Section 4

Design Overview

4.1 GSAS Design Overview

The GSAS processing system is designed to be both efficient and flexible. The system is designed for operational flexibility, considering data availability constraints and reprocessing requirements. In order to meet these requirements, the design of the software consists of four functional layers which work together to perform the data processing function. From the bottom up, the first layer is a set of generic library routines which form the foundation of the software. The second layer is comprised of the science algorithm subsystem libraries, which perform the actual transformation from raw data into GLAS products. The third layer is the subsystem managers, which control the execution of the science algorithms. The fourth and final layer is GLAS_Exec, a shell which surrounds the subsystem managers and provides standardized I/O, error handling, and initialization.

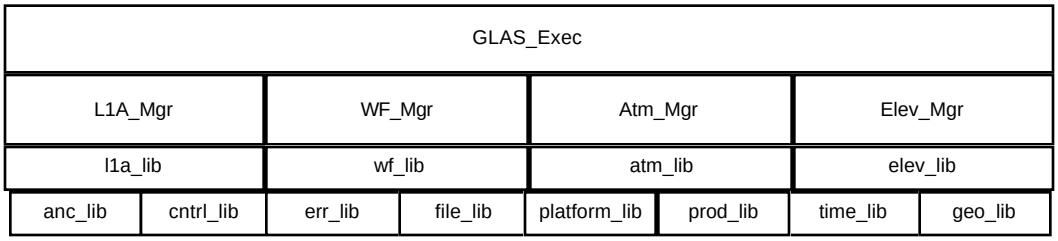

Figure 4-1 GSAS Layers

The generic library routines are grouped into foundation libraries by function. The libraries provide standardized services such as error handling, file open/close, keyword/value parsing, and time conversion, for example.

4.2 Files

Throughout this document, files are referenced as one of two types: GLA or ANC. GLA files are integer-binary format product files containing Level 0-2 GLAS data. The GLA files are fixed-length binary files containing scientific measurements. ANC files are multi-format ancillary files supplied by the science team which are required for processing. These files are detailed in the GLAS Data Management Plan and GLAS Data Product Users Guide.

4.3 Science Algorithms and Products

The science algorithms are published in Algorithm Theoretical Basis Documents (ATBD) provided by the GLAS Science Team. The resulting code is grouped into four ATBD subsystems separated by scientific discipline. These subsytems, science data products, and the science algorithm libraries listed in Table 4-1.

Table 4-1 Subsystem, Libraries and Products

Subsystem	Library	Output Products
L1A Processing	l1a_lib	GLA01-04
Waveform Processing	wf_lib	GLA05
Atmosphere Processing	atm_lib	GLA07-11
Elevation Processing	elev_lib	GLA06,12-15

The subsystems are designed such that data required by each subsystem is available from a product (data file) written by a preceding subsystem. As a result there is very little data dependence between the subsystems.

Associated with each ATBD subsystem is a corresponding Subsystem Manager. These Managers use control input to determine what processes to execute within the subsystem and what data to write.

4.4 GSAS Utilities

In addition to the main GLAS_Exec processor, there are several utilities which perform various data transformations and computations. These utilities use the same core library routines as GLAS_Exec. There will be two main types of utilities:

- Utilities executed infrequently – based on static or near-static input. Examples are:
  - Reference orbit groundtrack file creation
  - Create DEM file
  - Ingest and reformat geoid file
  - Create regional masks data set
  - Create surface type file
  - Create global and regional load tide grids
- Utilities executed routinely as part of daily production processing. Examples are:
  - Calculate granule start times and ascending node times
  - Create level 0 index files
  - Subset Met data files

- Create Browse products
- Create QA products

Other types of utilities may be added as the design and implementation progress. Each utility will be delivered with its own Detailed Design document.

Section 5

Foundation Libraries

The base level of GSAS software is implemented as a set of core libraries. These libraries are coded in a generic manner such that GLAS_Exec, GSAS Utilities, and other software can make use of the code. This design maximizes code reuse and its inherent advantages.

Library code is implemented in separate directories and grouped by functional area. A single makefile in each library directory will compile the code into a dynamically-linked shared library. A “master” makefile will compile all the libraries and create the final binaries in one step. See the GSAS Version Description document for details on file layout and compilation specifics.

There is a set of dependencies between the libraries. Order in which libraries are compiled is important since libraries may depend upon other libraries for support routines.This is not relevant if the developer uses the supplied “master” makefile, but the developer should be aware that these dependencies exist. This is illustrated in Table 5-1.

Table 5-1 Library Inter-dependencies

To build...	The following libraries are required...
platform_lib	<none>
time_lib	platform_lib
cntrl_lib	platform_lib
err_lib	platform_lib,
math_lib	platform_lib
anc_lib	platform_lib, cntrl_lib, err_lib, math_lib
prod_lib	platform_lib, cntrl_lib, err_lib
file_lib	platform_lib, cntrl_lib
geo_lib	platform_lib, cntrl_lib, err_lib, math_lib, anc_lib

5.1 The Platform Library (platform_lib)

platform_lib is the most basic library in the foundation libraries. Nearly all GSAS code uses routines from the platform library. The purpose is to provide consistent datatypes across all GSAS software, to provide a place for storing constants, and to

provide compiler-dependent F90 routines. Modules included in the platform_lib are described in Table 5-2.

Table 5-2 platform_lib Modules

Module	Description
kinds_mod	Defines the basic GLAS datatypes, for example 2 byte integers, 4 byte integers, 4 byte reals, and 8 byte reals.
types_mod	Defines common complex GLAS datatypes, including structures.
const_glob_mod	Defines common global constants. These constants are initialized as parameters or have values read from an ancillary file.
const_atm_mod	Defines atmosphere-related constants. These constants are initialized as parameters or have values read from an ancillary file.
const_elev_mod	Defines elevation-related constants. These constants are initialized as parameters or have values read from an ancillary file.
const_l1a_mod	Defines L1A-related constants. These constants are initialized as parameters or have values read from an ancillary file.
const_wf_mod	Defines waveform-related constants. These constants are initialized as parameters or have values read from an ancillary file.
lnblnk	Returns position of the last non-blank character in a string. Provided for those F90 implementation which do not support this function.
vers_platform_mod	Version information for the library.

5.2 The Control Library (cntrl_lib)

cntrl_lib provides control-related functions to GSAS software. Components include routines for parsing “keyword=value” formatted files, string functions, user-interface functions, and a common file control datatype. Modules included in the cntrl_lib are described in Table 5-3.

Table 5-3 cntrl_lib Modules

Module	Description
centertext_mod	Centers a text string within an 80 character padded string.
compare_kval_mod	Compares keyvalues against label. Strings are converted to uppercase before a comparison is performed. This ensures that keyvalues are not case-sensitive.
doubleline_mod	Prints an 80 character double line to the supplied IO unit.
fStruct_mod	Defines a generic GLAS file info structure. Also contains routines to initialize and print a file info structure.
getans	Reads a character of input, and validates that input from a list of acceptable values.

Table 5-3 cntrl_lib Modules (Continued)

Module	Description
keyval_mod	Defines a keyword=value datatype.
multimenu_mod	Returns a set of logicals based on user menu selection
parse_keyval_mod	Parses keyword and value components from argument string.
read_line_mod	Reads a line of input, skipping comments (#).
singleline_mod	Prints an 80 character single line
strcompress	Compresses multiple spaces to a single space within a string
strtrim	Trims white space from around a text string
tolower	Converts alpha characters to lower case
toupper	Converts alpha characters to upper case
writebanner_mod	Prints banner at start of processing
vers_cntrl_mod	Version information for the library.

5.3 The Error Library (err_lib)

err_lib provides status and error-related functions to GSAS software. err_lib is designed to read messages from an ancillary file. Errors and status messages (henceforth referred as errors) are reported to an output ancillary file (if available) and to standard output (stdout). Errors have negative numeric designations; status messages have positive designations. Errors are designed to be configurable as to the severity of the error and frequency of printout.

The error file consists of sections of independent numbers. Within a section, these numbers must be consecutive. An example of the utility of error sections is that GLAS_Exec has separate sections for itself and each of the subsystems. This eases maintenance of the error file. Section starts are hardcoded to the following: (absolute value): 0,10000,20000,30000,40000,50000,60000,70000,80000,90000. 0-9999 is reserved for GLAS_Error, itself

GSAS software is required to terminate processing only upon receipt of a fatal severity code. Only the highest-level routine may terminate processing, all other routines return the error code to their calling routine.

The structure of an ancillary error file is standard GLAS “keyword=value” format. The two fields are a keyword and an error string. The format defined in Figure 5-1.

```
KEYWORD=nnnnnnxttttttttttttttttttttttttttttttttttttttttttttttttttttxsxffffff
```

Figure 5-1 Error Ancillary File Format

The KEYWORD can have the value of “ERROR” or “STATUS”

The VALUE is a text string with the specific format defined in Table 5-4.

Table 5-4 Error String Format

Character	Positions	Description
n	1-6	Error designation (must be sequential within a section)
x	7,58,60	Space character (delimiter)
t	8-57	Short description
s	59	Error severity (0=no error, 1=informaton, 2=warning, 3=fatal)
f	61-66	Frequency of reporting (message is reported on 1st occurrence, then every f'th time)

Modules included in the err_lib are described in Table 5-5.

Table 5-5 err_lib Modules

Module	Description
ANC06_mod	Writes an error message to the ANC06 unit in a standard format. In order to avoid cyclic dependencies, ANC06_mod will not use GLAS Error_mod upon encountering an error (since GLAS Error WILL use ANC06_mod). A result code will be returned, but the caller must act upon it, if necessary.
ErrDefs_mod	Defines the GSAS error data structure.
ErrorBoot_mod	Initializes the error to generic values before the ancillary error file is read.
ErrorInit_mod	Perform initializations for the Error and Status function by extracting the error variables from argument error strings. This routine does dynamic array allocation so that the number of errors is not fixed. A routine is also provided to print the parsed errors.
GLAS_Error_mod	Receives an error number as an argument, looks up the error, writes the error to ANC06 and stdout, and returns a severity code to the calling process.
WriteError_mod	Formats an error and writes to ANC06 and stdout.
vers_err_mod	Version information for the library.

5.4 The Math Library (math_lib)

math_lib provides standard math routines to GSAS software. Components include bilinear interpolation and matrix multiplication. Modules included in the cntrl_lib are described in Table 5-6.

Table 5-6 math_lib Modules

Module	Description
c_bilin_interp_mod	Calculates the value of properties at a point by doing a bilinear interpolation of the 4 points straddling it.
c_matmul_mod	Returns the product of two matrices.
vers_math_mod	Version information for the library.

5.5 The Ancillary Library (anc_lib)

anc_lib provides routines to read and parse GLAS ancillary files. GSAS ancillary files are of various formats. The error ancillary file has been previously described.The constants ancillary files (ANC07) are in standard GSAS “keyword=value” format described in the cntrl_lib section. The specific ANC07 modules parse the returned text value using internal reads to store the result in integer of floating point format. Modules included in the anc_lib are described in Table 5-7.

Table 5-7 anc_lib Modules

Module	Description
anc01_met_mod	Reads meteorological (met) header data into a global data structure. Structures exist for two met header files. Also verifies the existence of associated met data files and provides a routine to write the met header information to stdout.
doubleline_mod	Prints an 80 character double line to the supplied IO unit.
anc07_atm_mod	Reads and parses atmosphere-related constants from a constants ancillary file.
anc07_glob_mod	Reads and parses global constants from a constants ancillary file.
anc07_elev_mod	Reads and parses elevation-related constants from a constants ancillary file.
anc07_err_mod	Reads and parses error constants from a constants ancillary file.
anc07_l1a_mod	Reads and parses L1A-related constants from a constants ancillary file.
anc07_stat_mod	Reads and parses status constants from a constants ancillary file.
anc07_wf_mod	Reads and parses waveform-related constants from a constants ancillary file.
anc08_pod_mod	Contains Precision/Predict Orbit Determination (POD) record length and a flag to determine if POD is of predicted or precision quality.

Table 5-7 anc_lib Modules (Continued)

Module	Description
anc09_pad_mod	Contains Precision Attitude Determination (PAD) record length, public data structure, availability flag, and routines to initialize and read PAD records.
anc12_dem_mod	Contains Digital Elevation Model (DEM) record lengths, unit number, public LandMask, and routines to read, calculate and print the DEM values.
an c13_geoid_mod	Contains the Geoid record length, public grid and routines to initialize and read the geoid.
anc16_ltide_mod	Contains the record length and unit of the load tide ancillary file.
anc17_otide_mod	Contains the record length and unit of the ocean tide ancillary file.
anc18_stdatm_mod	Reads and stores the standard atmosphere ancillary file.
anc24_rot_mod	Contains the record length and unit of the rotation matrix ancillary file.
anc30_aer_mod	Reads and stores the global aerosol map ancillary file.
anc31_trop_mod	Reads and store the global aerosol trop map ancillary file.
vers_anc_mod	Version information for the library.

5.6 The File Library (file_lib)

file_lib provides standard routines to open and close GSAS files using the passed file info structures. Modules included in the file_lib are described in Table 5-8.

Table 5-8 file_lib Modules

Module	Description
OpenFInFile_mod	Opens an input file.
OpenFOutFile_mod	Opens an output file.
vers_file_mod	Version information for the library.

5.7 The Time Library (time_lib)

time_lib is the only GSAS source code implemented in C. It is an implementation of a GSFC time library and used by GSAS with little to no modification. time_lib provides

routines for converting to/from various time formats. Modules included in the time_lib are described in Table 5-9.

Table 5-9 time_lib Modules

Module	Description
dateinterface	Has routines for the following functions: -add two arrays holding times into a third array -add a yymmdd and a day -add a yyyymmdd and a day -find the difference between two yymmdd's in days and seconds -find the difference between two yyyymmdd's in days and seconds -convert between yyyymmdd, hms, mjd, fday, and mjdsec -convert yymd fday to J2000 days and fday -convert J2000 days and fday to yymd fday -convert yymmdd or yyymmdd to yyyymmdd -convert yyyymmdd to yymmdd -convert mjd to yymmdd -convert mjd to yyyymmdd -convert yymmdd to mjd -convert yyyymmdd to mjd -convert hhmmss to fday -convert fday to hhmmss -convert fday to hm with decimal seconds -convert yyyymmdd to ddd -convert yyyymmdd to yyyyddd -convert yyyyddd to yymmdd -convert mjd to mjdsec -convert mjdsec to mjd -convert mjdsec to sec -check if yyyy is a leap year
vers_file_mod	Version information for the library.

5.8 The Product Library (prod_lib)

prod_lib provides routines to read, write, and convert GLAS products. The routines (and concepts) are fully described in the GLAS_Exec section. Modules included in the prod_lib are described in Table 5-10 (where *xx* = a GLAS product number [01-15]).

Table 5-10 prod_lib Modules

Module	Description
GLA00_mod	Contains routines for reading GLA00 APIDs and Index file.
GLA00_prod_mod	Contains public data structures for GLA00 APIDs and routines to initialize and print the data structure.
GLA*xx*_mod	Contains routines for reading and writing GLA*xx* product data structures.
GLA*xx*_alg_mod	Contains public data structures for GLA*xx* algorithmdata and routines to initialize and print the data structure.

Table 5-10 prod_lib Modules (Continued)

Module	Description
GLA*xx*_prod_mod	Contains public data structures for GLA*xx* product data and routines to initialize print the data structure.
GLA*xx*_scal_mod	Contains public data structures for GLA*xx* scale data and routines to initialize and print the data structure. Also contains routines to convert from product units to algorithm units and the reverse.
GLA*xx*_Pass_mod	Passes common data from a lower-numbered product/algorithm data structure to higher-numbered product/algorithm data structures.
GLA*xx*_flags_mod	Contains routines for packing and unpacking GLA*xx* flags.
common_flags_mod	Contains routines for packing and unpacking common flags.
conversions_mod	Contains routines for performing common data conversions.
prod_def_mod	Contains record sizes for all GLA*xx* products.
vers_file_mod	Version information for the library.

Section 6

GLAS_Exec Processes

6.1 Function

The topmost level of the GSAS data processing software is GLAS_Exec. This process is responsible for controlling the data processing. It performs initializations, setting constants, reading ancillary data, data input and output, and global error handling. It also performs a synchronization function for processing the data granules.

GLAS_Exec executes processes for the four major subsystems: L1A, Waveforms, Atmosphere, and Elevation. These subsystems contain implementations of the algorithms used to process GLAS data. GLAS_Exec is the superstructure surrounding the subsystems structures.

6.2 Design Constraints/Decisions/Assumptions

- The system start and stop will be controlled by the main process, GLAS_Exec.
- Processing will be done a record at a time, though individual subsystems may buffer multiple records before processing.
- Control flags will determine which subsystem or subsystem process will be executed.
- GLAS_Exec will time synchronize the input data.
- The system will provide for partial processing and reprocessing scenarios

6.3 Product Internal Data Storage, Conversion and I/O

The GLAS_Exec I/O and unit conversion process is sufficiently complex and important to describe in detail.The design of this process is what allows GLAS_Exec to meet the reprocessing requirements.

First, some definitions: (1) algorithm data (in units for algorithm use) are that data which are in a form most favorable for display and calculation; (2) product data (in units for I/O) are data which are in a form most favorable for machine independence and storage efficiency. It is important to understand the process by which algorithm data gets transformed into product data and vice vera.

6.3.1 Product Modules

There are several different types of modules involved in the product conversion process. These modules were briefly described in the **prod_lib** section but will be detailed here. Table 6-1 (where xx = a GLA product number [01-15]) defines each

component. All modules are designed with software reuse as a primary goal. The product modules are already being used in utility software outside of GLAS_Exec.

Table 6-1 Product Module Functionality

Module	Functionality
kinds_mod	defines basic data types (4-byte integer, 8-byte real, etc.)
types_mod	defines any global data structures
GLA*xx*_prod_mod	defines product-specific (where xx=product number) record format and associated global product data structure. Each module also includes one subroutine to initialize the product data and another to print the data in a human-readable form.
GLA*xx*_mod	contains routines to read (ReadGLAxx) and write (WriteGLAxx) the product data structure in binary format.
GLA*xx*_alg_mod	defines product-specific global algorithm data structure. Each module also includes one subroutine to initialize the algorithm data and another to print the data in a human-readable form
GLA*xx*_scal_mod	defines product-specific global scaling data structure. Also includes subroutines to initialize the scaling data, convert from product to algorithm format (GLAxx_P2A), convert from algorithm to product format (GLAxx_A2P), and print the scaling data in a human-readable form.
common_flag_mod	contains routines for packing/unpacking common flags.
GLA*xx*_flag_mod	contains routines for packing/unpacking product-specific flags.

6.3.2 Internal Product Data Storage

Data for each product are stored internally in two different formats. For each product, there is one global data structure containing product data. These data are in the exact same format as the integer-binary data written to and read from output files. There is also a global data structure for each product containing algorithm-format (mostly double precision) data for use in scientific calculations. The product modules and the GLAS_Exec Managers use these public data structures. However, data are passed from the Managers to the science algorithms via the argument list.

6.3.3 Invalid Values

Not all data received from GLAS will be suitable for science processing. In addition, given the nature of the raw telemetry packets, some data may be missing. The concept of an “invalid value” is used to signify that data is invalid or missing and should not be used for processing. Invalid values are datatype-specific values which are defined in the GLAS global constants module. The current values are defined in Table 6-2.

Table 6-2 Invalid Values

Datatype	Invalid Value
1 byte integer	127
2 byte integer	32767
4 byte integer	2147483647
4 byte real	x7F7FFFFF
8 byte real	x7FEFFFFFFFFFFFFF

6.3.4 Product Input/Output

GLAxx product files are defined as integer-binary fixed-length files. These product files will contain multiple text header records with the same length as the data record. The headers are not yet implemented and will be described in an appropriate version of this document.

The GLAxx_prod_mod defines a specific data structure which exactly matches the format of each data record of the appropriate product file. This data structure is used in an unformatted direct-IO statement to read / write a data record from / to disk.

When multiple products are read simultaneously, a data record from a lower-numbered product is read before the data from a higher-numbered product. This is important to the concept of “Pass-thru” (explained in Section 6.3.6).

6.3.5 Product-to-Algorithm Conversion (P2A)

When a data record is read from disk into memory, the data are stored in the product data structure. In order to be useful in scientific calculations, the data must be converted from product format into algorithm format. The process is called “Product-to-Algorithm Conversion”.

When a record of data is read, the values are stored in a product data structure. The appropriate algorithm data structure is initialized to either zeros or invalid values, as specified by the product documentation.

Each product variable is checked for an invalid value. If the data is determined to be invalid, no conversion is performed. As a result of initializing the algorithm structure appropriately, if the product variable is invalid, the algorithm value, by default, contains an invalid value.

If the values are determined valid, the data will be converted from product to algorithm format by one or more of the following processes.

- converting to unsigned (if necessary)
- scaling by a scale factor:
  Algorithm_Value = Product_Value Scale_Factor*
- unpacking bits into individual flags.

For the most part, scaling is performed by multiplying the integer product value by a floating point scale factor and storing the result in a double-precision algorithm variable within the global algorithm data structure. The exceptions to this rule are flags, which are unpacked with specific subroutines and a few variables which are used as integers by the science algorithms.

6.3.6 Pass-Thru

After a product is read and converted to algorithm format, common data must be passed from lower-numbered product / algorithm data structures to higher-numbered product / algorithm data structures. This pass-thru process enables re-processing to be treated the same as normal processing. It is important that both product and algorithm data is passed. The subsystem managers (discussed below) are designed to take full advantage of the pass-thru process.

6.3.7 Subsystem Managers

The subsystem managers ‘use’ the global algorithm data structures. If an intermediate conversion is necessary, the managers create local variables. The managers pass the appropriate variables to the science algorithms via the argument list. (L1A is an exception to this since the L1A routines basically use the entire data structures.) Specific algorithms are executed based on the state of control flags received from GLAS_Exec in order to allow for re-processing

.A key concept is that the manager uses the variable in the highest-numbered product for which it is responsible. For example, if the same variable is on GLA05 and GLA06, the elevations manager always uses the variable from the GLA06 algorithm structure, no matter if GLA06 is read for input or not. The pass-thru process ensures that the value is always there.

After a science algorithm returns execution to the manager, the manager performs its own pass-thru function. It copies any local variables back to the algorithm data structure and then passes any modified algorithm variables to the higher-numbered product / algorithm data structures. This is essentially a repeat of the pass-thru process described in 6.3.6, except the candidate variables are limited to those modified by each respective science algorithm.

6.3.8 Algorithm to Product Conversion (AP2)

After the manager has finished executing science algorithms, each algorithm structure must be converted back to product data. This is essentially a reverse of the P2A process.

First, the product structure is initialized. Then, each algorithm variable is checked for an invalid value. If the variable is determined valid, the data will be converted from algorithm to product format by one or more of the following processes.

- unscaling by a scale factor:
  Product_Value = nint(Algorithm_Value/Scale_Factor)
- unpacking bits into individual flags.

For the most part, scaling is performed by taking the nearest integer of the double precision algorithm value divided by a floating point scale factor. The result is stored back into an integer product variable within the global product data structure. The exceptions to this rule are flags, which are packed with specific subroutines and a few variables which are used as integers by the science algorithms.

6.4 Control and Constants Files

GLAS_Exec will use a text-based keyword=value format for all control and constants (ANC07 files). The control files will be sectioned such that the GLAS_Exec-specific portion can reside inside a larger, more generic SMDS control file. Standardization on text-based keyword=value files allows for good software reuse and greater consistency for the software user.

6.5 DFDs and their Descriptions (1.4)

The GLAS_Exec top-level data flow diagram shows the major components of GLAS_Exec.

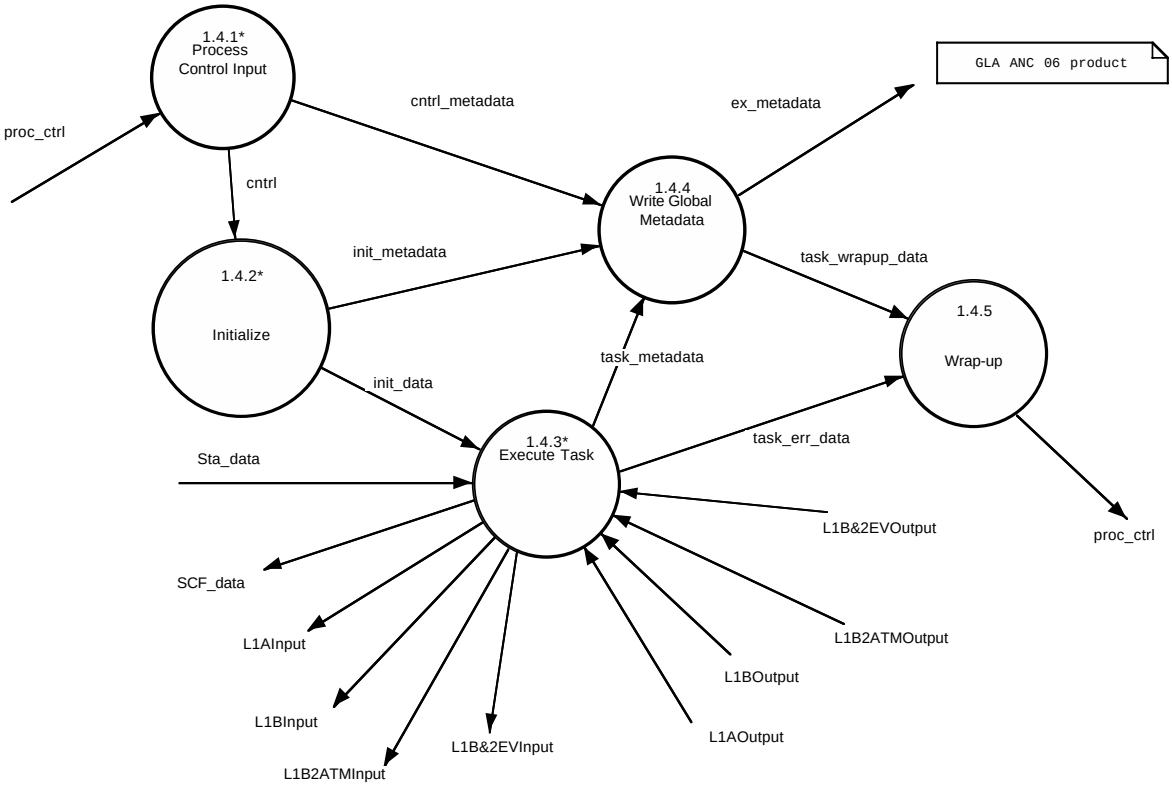

Figure 6-1 GLAS_Exec Top DFD

6.5.1 Process Control Input (1.4.1)

This process will read and parse a keyword=value format control file to determine which subsystems processes will be executed. It will determine the input files to use and the output files that will be produced. It will generate an error if improper file-name or inconsistent processes are specified and will validate the start and stop times for the data to be processed. The control data will pass to the submanagers to determine the execution path.

Due to reprocessing requirements, GLAS_Exec is mostly a state machine. The control file may specify which of the GLA files to use as input, what processes will transform the input data, and what files are to be written as output. All file naming is performed in the Scheduling and Data Management System (SDMS) and the names of all input, output, and ancillary files are identified in the control file. All parsed control input, as well as things such as program name and version are written as metadata to ANC 06. The contents of the control file are fully described in Appendix C.

6.5.2 Initialize (1.4.2)

The Initialize process will initialize all data structures required for processing based on control data that is specified. These structures include product and algorithm structures and also input and output file structures. This process will also be responsible for opening all input and output files that are required for a particular processing scenario. It will open and read the required ancillary and constants files.

Only those ancillary files whose data are determined to be efficiently kept in core are read in at this time. All requested ancillary files are opened here. While subsystem processes may independently read ancillary data files, they will not open and close files themselves. All initialization and parsed control data are written as metadata to ANC06.

6.5.3 Execute Task (1.4.3)

The Execute Task process executes the science algorithms specified by ATBDs for each of the four subsystems. The process takes data one record at a time and uses control parameters to determine which subsystem and which processes to execute within the subsystem. The process passes the data to the appropriate subsystem manager for transformation in lower level processes. The subsystem managers write out the output in files specified by the control input.

If there is a fatal error in any of the subsystems, the execution is interrupted and an error code is returned to GLAS_Exec, which is the only process from which the system can be exited.

6.5.4 Write Global Metadata (1.4.4)

The Write Global Metadata process is responsible for recording data about the processing from each of the subsystems. It records the processes that are executed, the error messages that occur and the versions of the software that is running. It gives information about the quality assurance files that are produced during an execution of GLAS_Exec.

6.5.5 Wrap-up (1.4.5)

Wrap-up records data that indicate that the execution of GLAS_Exec is complete. It writes data that indicate what files are closed and which data granules have been processed. It triggers the calculation of the final statistics from all the subsystems and writes them to the quality assurance files.

6.6 GLAS_Exec Decompositions

6.6.1 Process Control Input (1.4.1)

The Process Control Input decomposes into the following subprocesses:

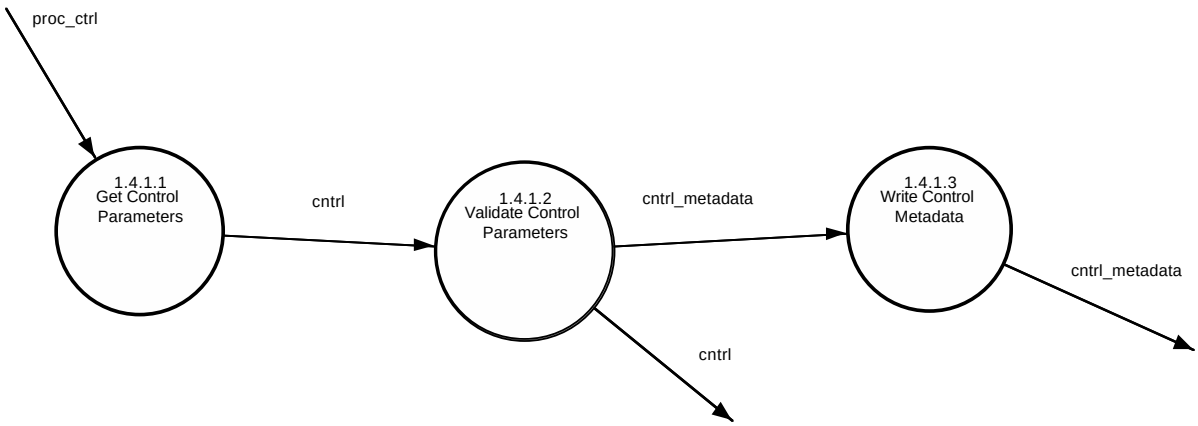

Figure 6-2 Process Control Input

6.6.1.1 Get Control Parameters (1.4.1.1)

The Get Control Parameters process will read a control file, parse it and extract values from a keyword-value pair in order for GLAS_Exec to determine the parameters and scenarios to be used in processing.

6.6.1.2 Validate Control Parameters (1.4.1.2)

The Validate Control Parameters process will examine the parameters for consistency. It will check that the correct input files are used for a particular execution of GLAS_Exec and the processing interval for processing a granule.

6.6.1.3 Write Control MetaData (1.4.1.3)

The Write Control Metadata process writes control information to the metadata file, ANC06.

6.6.2 Initialize (1.4.2)

The Initialize process decomposes into the following subprocesses:

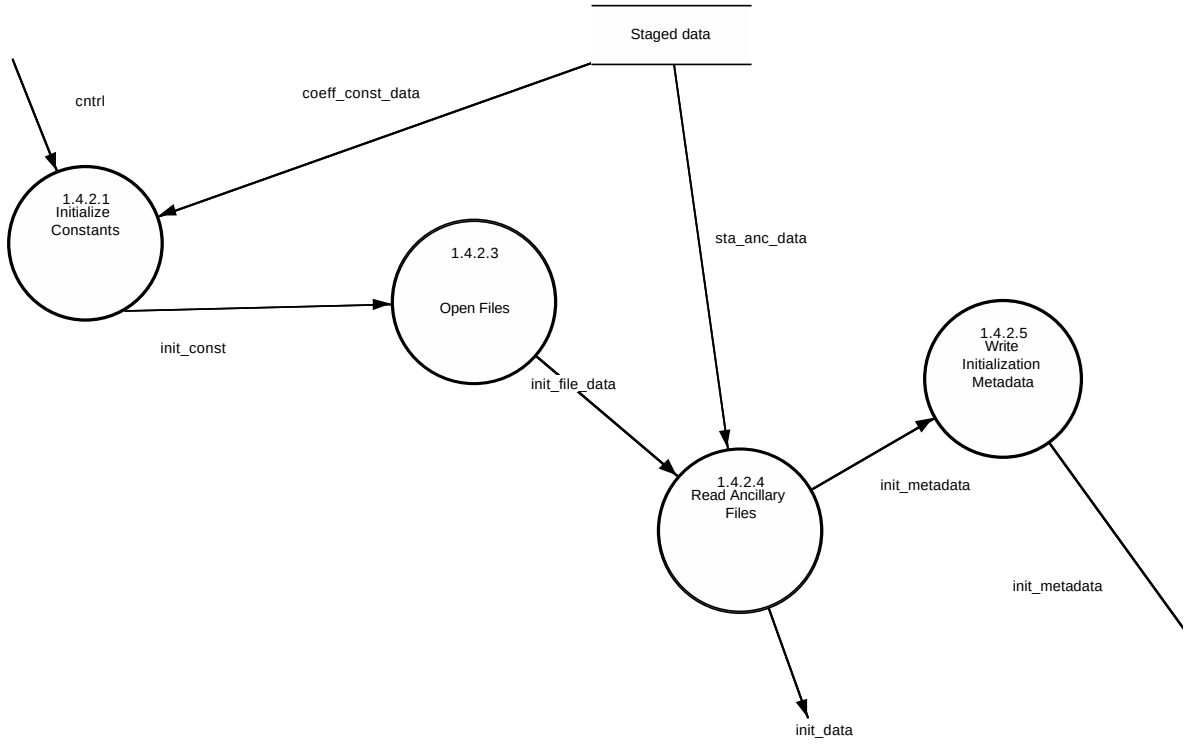

Figure 6-3 Initialize

6.6.2.1 Initialize Constants (1.4.2.1)

The Initialize Constants process retrieves all the required constants filenames specified in the control file and places them in data structures. Each subsystem will require a specific file together with a global constant file and an error definition file.

6.6.2.2 Open Files (1.4.2.3)

Opens Files opens all input and output files specified in the control file, storing the file handles in a file structure that GLAS_Exec initializes for access by the subsystem managers.

6.6.2.3 Read Ancillary Files (1.4.2.4)

This process reads the appropriate ancillary files specified for processing and parses the input to retrieve the constants. Only those ancillary files which are held in core are read at this point.

6.6.2.4 Write Initialization Metadata (1.4.2.5)

This process adds metadata to the ANC06 file.

6.6.3 Execute Task(1.4.3)

This process is responsible for running the subsystem managers based on information received from a control file. The Subsystem Managers control the top-most level of each subsystem. These managers are responsible for the execution of specified sub-ATBDs, the creation of appropriate GLA_SCF data products, and any post-record wrap-up. The managers are also responsible for the transfer of non-computed data (i.e., pass-through data) from one SCF product to another.

Pass-thru parameters are moved to the appropriate subsequent product to facilitate the abstraction of processing and reprocessing code. In other words, during the execution of the Subsystem Manager, parameters are moved from a sequentially lower numbered product to a higher numbered product prior to the sub-ATBD execution, which creates data for the higher numbered product. Pass-thru data which is placed on the higher numbered product is used in the call in the sub-ATBD rather than from the lower-numbered source product. This allows the sub-ATBD to be coded such that it does not matter if it is being run in a processing or re-processing scenario.

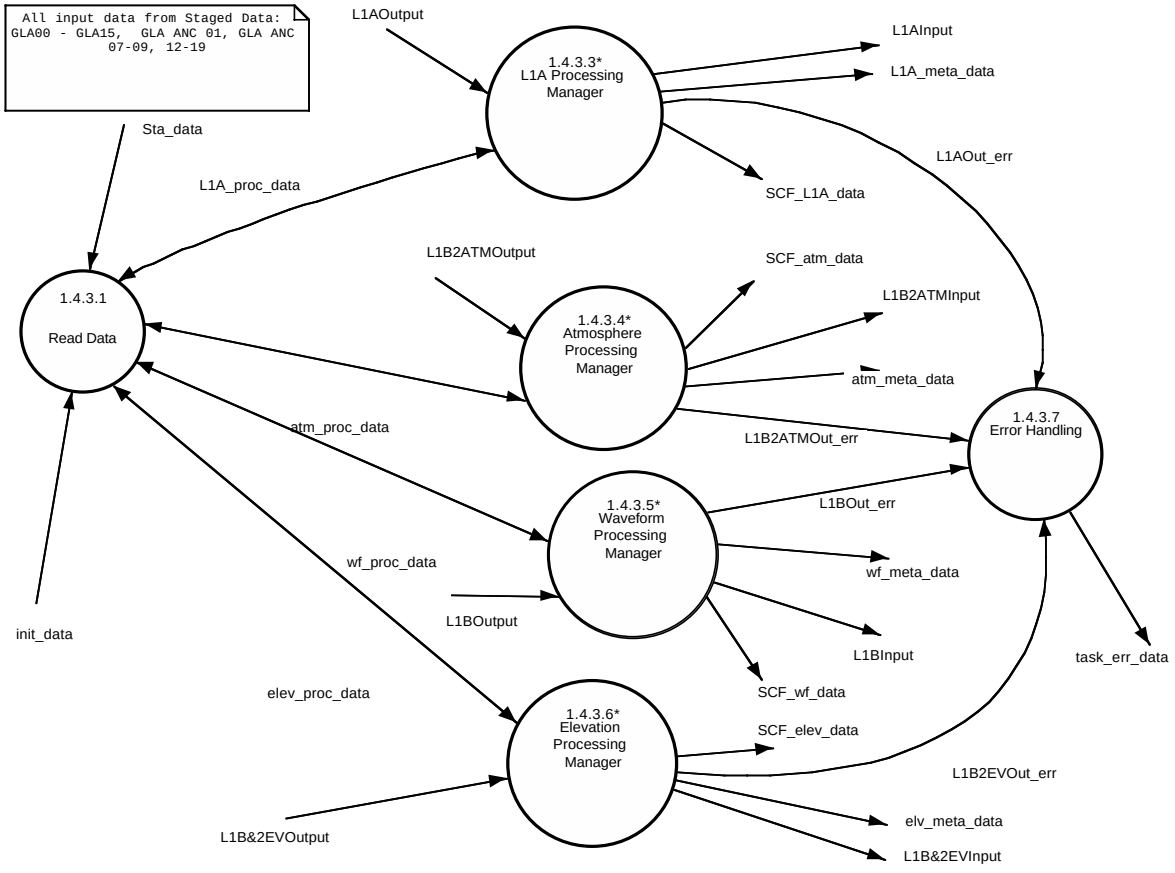

Figure 6-4 Execute Task

6.6.4 ReadData (1.4.3.1)

The ReadData process retrieves data on a record by record basis from the input files specified for processing. The files to be read are determined by the control input and the data read until all input products reach end-of-file. Control Flags will signify what data will be read. Availability flags will signify when a particular type of data is available. Data are time-synchronized upon input.

6.6.4.1 L1A Processing Manager (1.4.3.3)

This process constitutes the Level 1A subsystem that converts raw satellite data into engineering units. It is responsible for handling 1 sec records and converting them into Level 1 products consisting of engineering and science data. This process executes the algorithms specified by the algorithm Theoretical Basis Documents(ATBDs) produced by the Science team.

6.6.4.2 Waveform Processing Manager (1.4.3.4)

The Waveform manager process generates waveform-based information required to produce elevation products and output waveform characteristics. The process will include all calculations involving the waveform and other instrument parameters which are required for characterizing the surface or calculating the instrument range. The process will also create quality parameters to enable product quality determination. The process implements the ATBD specified by the science team

6.6.4.3 Atmosphere Processing Manager (1.4.3.5)

The atmosphere manager process receives 1 sec data and creates atmosphere parameters for the standard data products. The process also creates metadata and quality assessment data. The process is responsible for producing backscatter profile, Layer heights for aerosol and cloud layers, Cross sections and Optical depths products. The process implements the ATBD specified by the Science team.

6.6.4.4 Elevation Processing Manager (1.4.3.6)

The Elevation manager process generates the elevation standard data products. It creates parameters for a Level 1B time-ordered global product with a geodetically corrected surface-independent standard elevation, elevation corrections specific to each typo of surface: ice sheet, sea ice, land, and ocean; and geodetic corrections for each observation. From the global product four region-specific Level 2 elevation products, one each for ice sheet, sea ice, land, and ocean regions.

6.7 Submanager Decomposition

6.7.1 L1A Processing Manager (1.4.3.3)

The L1A Processing Manager process decomposes into the following subprocesses:

6.7.1.1 L1A Mgr Local Initialization (1.4.3.3.1)

The Local Initialization process performs any computations or initializations necessary before the ATBD processes are executed.

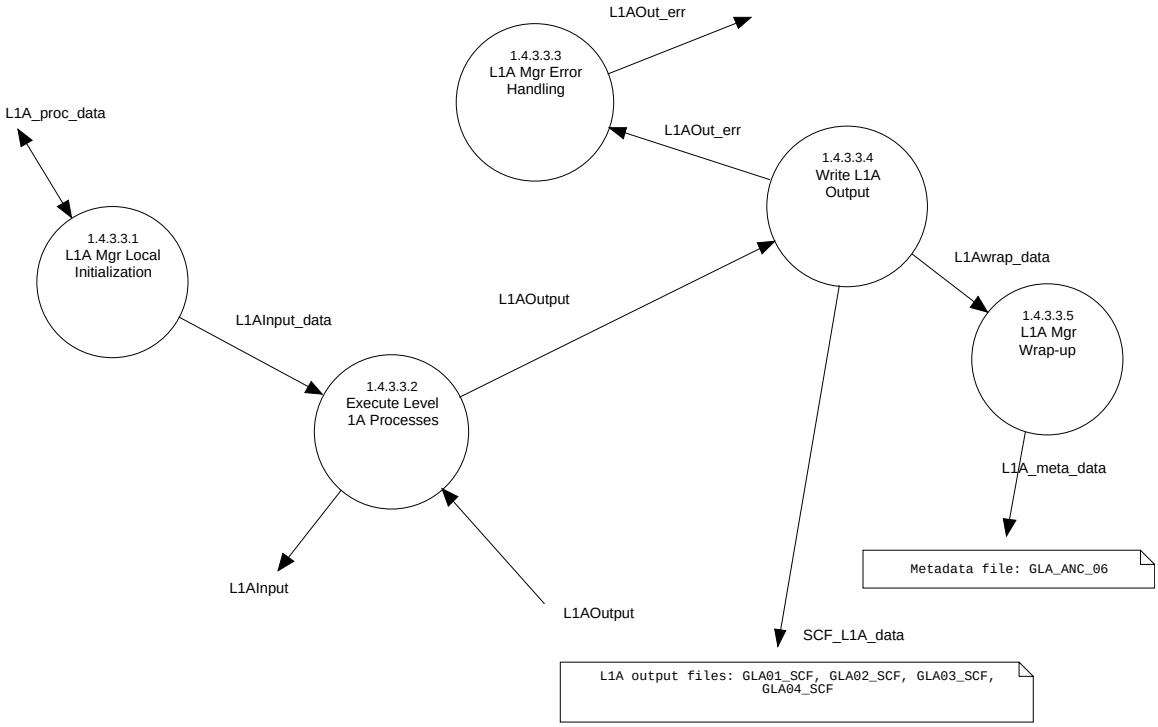

Figure 6-5 L1A Processing Manager

6.7.1.2 L1A Computations (1.4.3.3.2)

See section on L1A Computations

6.7.1.3 L1A Error Handling (1.4.3.3.3)

The Error handling process returns an error severity code to its upper level process.

6.7.1.4 Write L1A Output (1.4.3.3.4)

The Write L1A Output process writes the GLA01-GLA03 (GLA04, SRS, will be implemented in V2) products, which represent engineering, altimetry, and atmosphere data. It calls routines to convert algorithm data to product data before writing the product.

6.7.1.5 L1A Wrap-up (1.4.3.3.5)

The L1A Wrap-up process writes metadata to the ANC06 file and quality assurance data to specified QA files for the granule that was processed.

6.7.2 Atmosphere Processing Manager (1.4.3.4)

The Atmosphere manager performs local initializations and executes the ATBD processes. The manager creates 1 second GLA07 records from GLA02 data and is respon-

sible for buffering 20 seconds of data and writing five four second records to GLA08-11. This process performs wrap-up functions and writes quality assurance statistics.

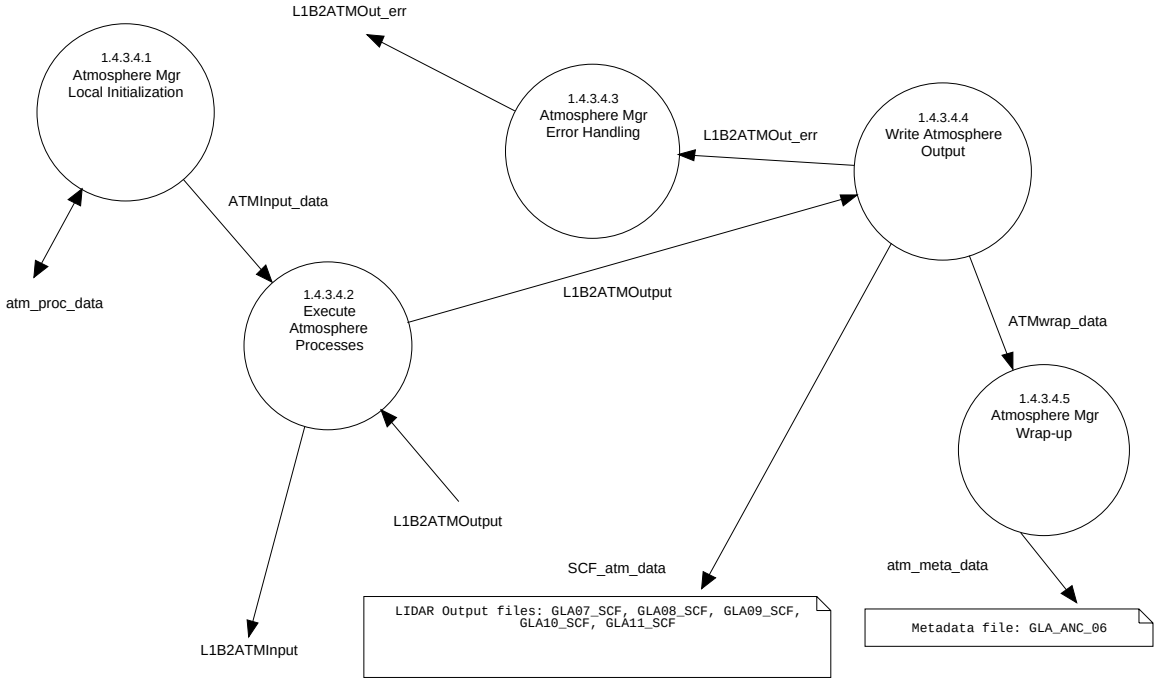

Figure 6-6 Atmosphere Processing Manager

Due to the low abundance of aerosols in the upper portion of the atmosphere, the aerosol data above 20 km need to be collected over 20 seconds to ensure a significant signal. Since the aerosol detection algorithm relies on the cloud detection algorithm and the optical properties algorithms rely on the aerosol detection algorithm, data for all these routines are buffered together at 20 seconds to ensure time synchronization with each other.

Similarly, the cloud and aerosol data under 20 km are collected for 4 seconds. Therefore, the cloud product (GLA09), the aerosol product (GLA08), and the optical properties products (GLA10 and GLA11) are written at once per 4 seconds, 5 records at a time for each 20 second buffer.

The buffering is based on time for a contiguous 20 seconds. Therefore, time gaps in the data are reflected in gaps in the buffer. If no data are available during the 20 second time period, then the buffer is empty. Algorithms that use the buffered data only process at times where sufficient data are present.

6.7.2.1 Atmosphere Mgr Local Initialization (1.4.3.4.1)

The Local Initialization process performs any computations or initializations necessary before the ATBD processes are executed.

6.7.2.2 Atmosphere Computations (1.4.3.4.2)

See section on Atmosphere Computations.

6.7.2.3 Atmosphere Error Handling (1.4.3.4.3)

The Error handling process returns an error severity code to its upper level process.

6.7.2.4 Write Atmosphere Output (1.4.3.4.4)

The Write Atmosphere Output process writes the GLA07-GLA11 products, which represent backscatter profiles, aerosol and cloud layers, cross sections and optical depth data. It calls a routine to convert algorithm data to product data before writing the product.

6.7.2.5 Atmosphere Wrap-up (1.3.3.4.5)

The L1A Wrap-up process writes metadata to the ANC06 file and quality assurance data to specified QA files for the granule that was processed.

6.7.3 Waveforms Processing Manager (1.4.3.5)

The L1A Processing Manager process decomposes into the following subprocesses.

6.7.3.1 Waveforms Mgr Local Initialization (1.4.3.5.1)

The Local Initialization process performs any computations or initializations necessary before the ATBD processes are executed.

6.7.3.2 Execute Waveform Processes (1.4.3.5.2)

See section on Waveform Computations

6.7.3.3 Waveform Error Handling (1.4.3.5.3)

The Error handling process returns an error severity code to its upper level process.

6.7.3.4 Write Waveforms Output (1.4.3.5.3)

The Write Waveform Output process writes the GLA05 output product, which represents waveform characteristics and Corrections. It calls a routine to convert algorithm data to product data before writing the product.

6.7.3.5 Waveform Wrap-up (1.4.3.5.4)

The Waveform Wrap-up process writes metadata to the ANC06 file and quality assurance data to specified QA files for the granule that was processed.

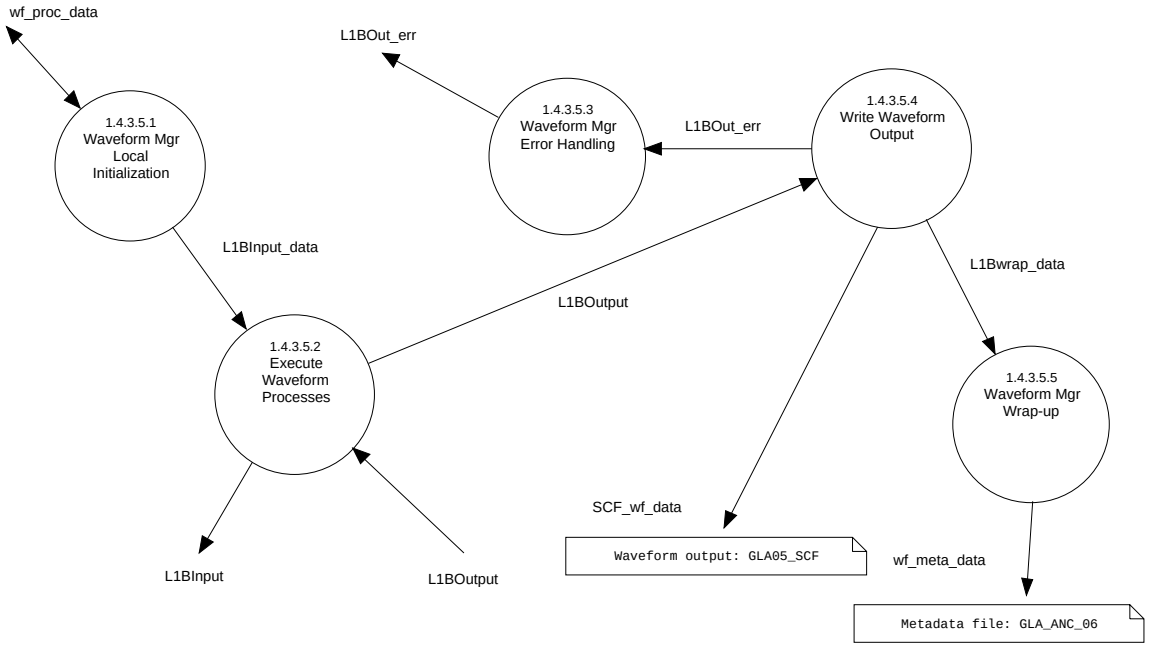

Figure 6-7 Waveform Processing Manager

6.7.4 Elevation Processing Manager (1.4.3.6)

The Elevation Processing Manager performs local initialization, executes the elevation ATBD algorithms, writes the elevation data in GLA12-15. It also executes a wrap-up process to write quality assurance statistics.

6.7.4.1 Elevation Mgr Local Initialization (1.4.3.6.1)

The Local Initialization process performs any computations or initializations necessary before the ATBD processes are executed.

6.7.4.2 Elevation Computations (1.4.3.6.2)

See section on Elevation Computations

6.7.4.3 Elevations Error Handling (1.4.3.6.3)

The Error handling process returns an error severity code to its upper level process.

6.7.4.4 Write Elevation Output (1.4.3.6.4)

The Write Elevation Output process writes the GLA12-GLA15 products, which represent region specific products for ice sheet, sea-ice, land and ocean data. It calls a routine to convert algorithm data to product data before writing the product.

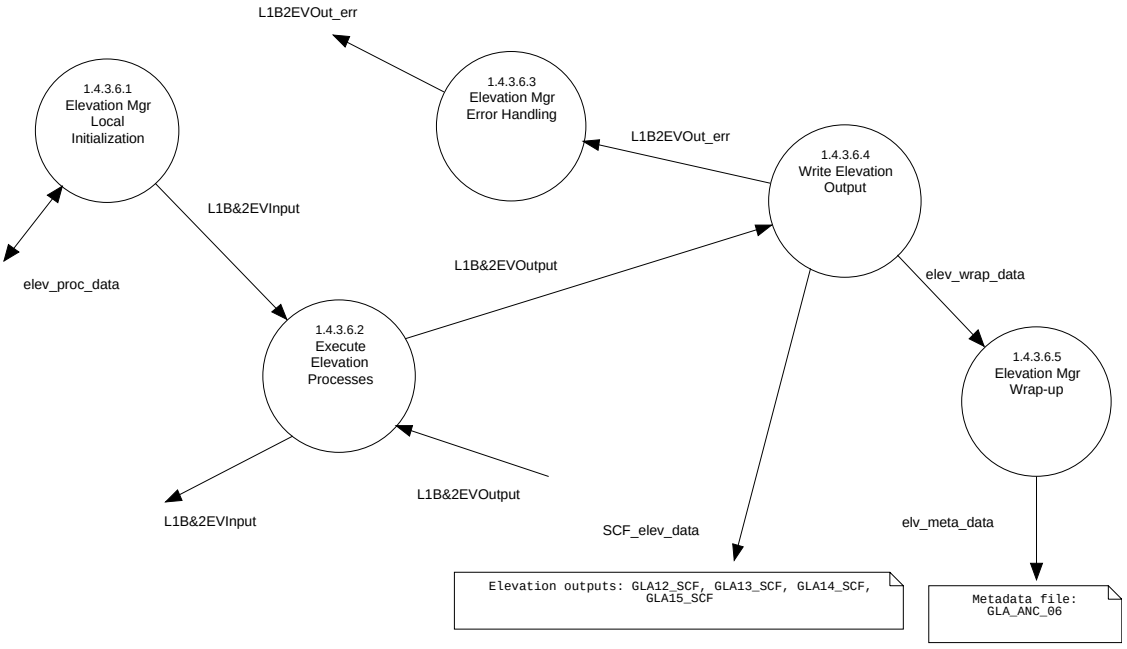

Figure 6-8 Elevation Processing Manager

6.7.4.5 Elevation Wrap-up (1.3.3.6.5)

The Elevation Wrap-up process writes metadata to the ANC06 file and quality assurance data to specified QA files for the granule that was processed.

Section 7

Level 1A Computations Processes

7.1 Function

This subsystem will create GLAS Level 1A data from the Level 0 GLAS instrument data products. The Level 1A Computations will input one second of Level 0 data and output one second of L1A data. Figure 7-1 illustrates the processes that comprise the subsystem.

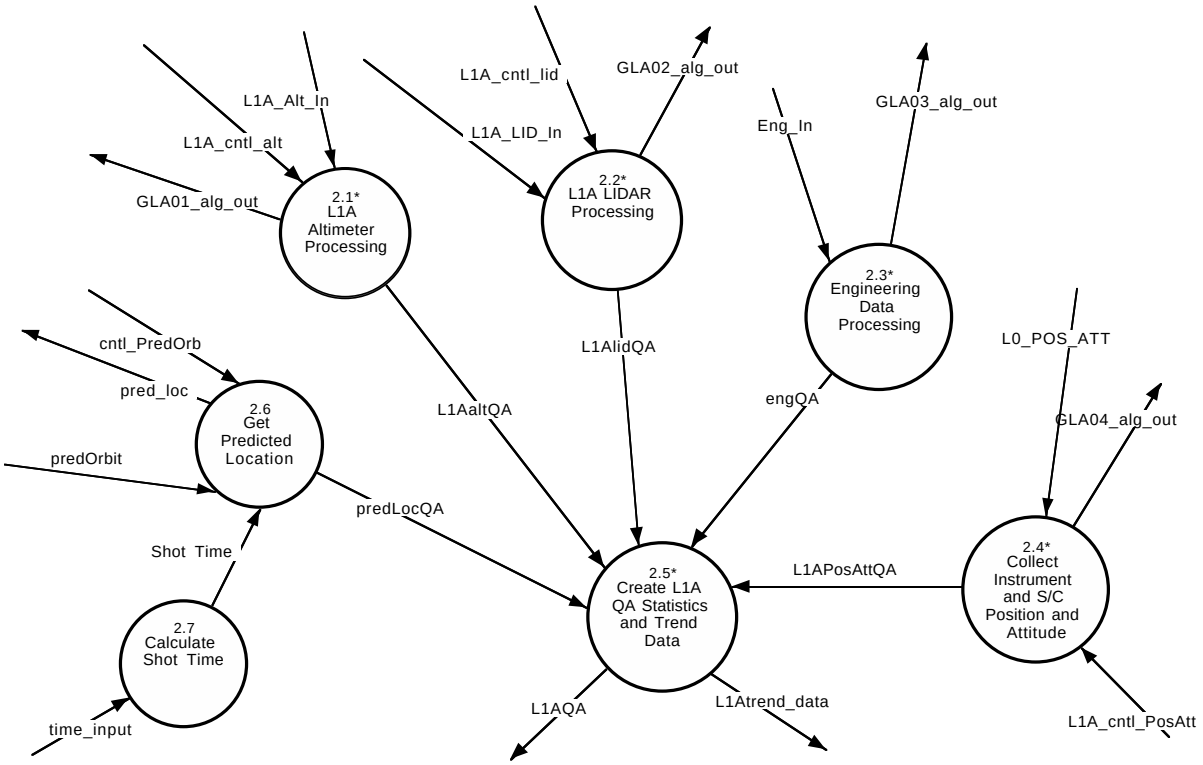

Figure 7-1 Level 1A Computations

7.2 Version 1 Design Decisions and Assumptions

The following design decisions were made:

- Implement only the processing to create the Level 1A products GLA01, GLA02, and GLA03.
- Monitoring, Trend, and QA processing will not be implemented.

The following assumptions were made:

- there will be no missing or bad data
- shots will occur at regular 25 millisecond intervals

- the once per second time is provided in a global variable

7.3 DFDs and their Descriptions

7.3.1 Level 1A Altimeter Processing

The purpose of the Level 1A Altimeter Processing (process 2.1) is to generate the data to be stored on the Level 1A Altimeter Data product (GLA01). The decomposition of process 2.1 is shown in Figure 7-2 "Level 1A Altimeter Processing (L_Alt)". This pro-

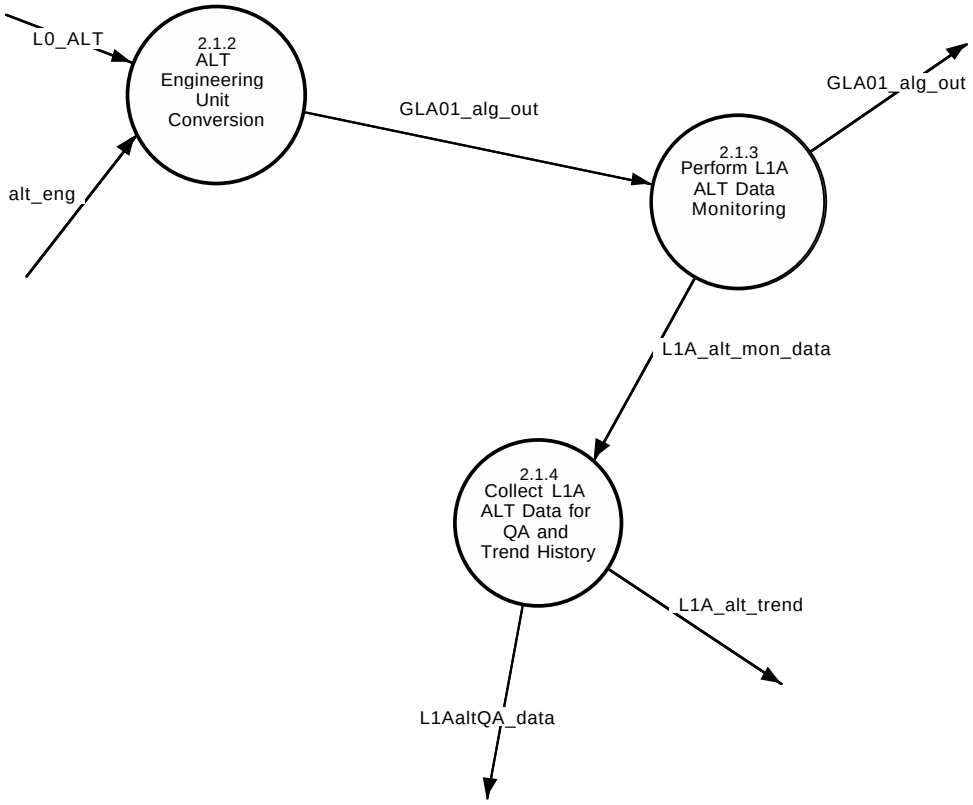

Figure 7-2 Level 1A Altimeter Processing (L_Alt)

cess converts the Level 0 altimetry data to engineering units; checks for data out of limits; and collects data for QA statistics and trend history. The subprocesses shown in Figure 7-2 are described in the following bullets.

- ALT Engineering Unit Conversion (L_alteuc) - subprocess 2.1.2 performs engineering unit conversion on the raw Level 0 altimetry data (L0_alt_data) to obtain the Level 1A altimetry data in engineering units which are stored in the output structure GLA01_alg_out. Any engineering / housekeeping data that are required to be on the GLA01 data product are collected here and placed in the output structure.

- Perform L1A ALT Data Monitoring (L_altmon) - subprocess 2.1.3 compares the Level 1A altimetry data in GLA01_alg_out to preset monitoring limits and thresholds. The data required for QA and trend history are output to subprocess 2.1.4 in the L1A_alt_mon_data structure. Flags are set appropriately in the L1A_alt_mon_data structure to indicate monitoring results.
- Collect ALT Data for QA and Trend History (L_altqatrnd) - subprocess 2.1.4 takes the data generated during the 2.1.2 and 2.1.3 subprocesses and updates QA (L1AaltQA_data) and trend (L1A_alt_trend) data.

7.3.2 L1ALIDAR Processing

The purpose of the L1A LIDAR Processing (process 2.2) is to generate the data to be stored on the Level 1A Atmosphere Data product (GLA02). The decomposition of process 2.2 is shown in Figure 7-3 "L1A LIDAR Processing (L_Atm)". This process converts the Level 0 data to engineering units; checks for data out of limits; and collects data for QA statistics and trend history. The subprocesses depicted in Figure 7-3

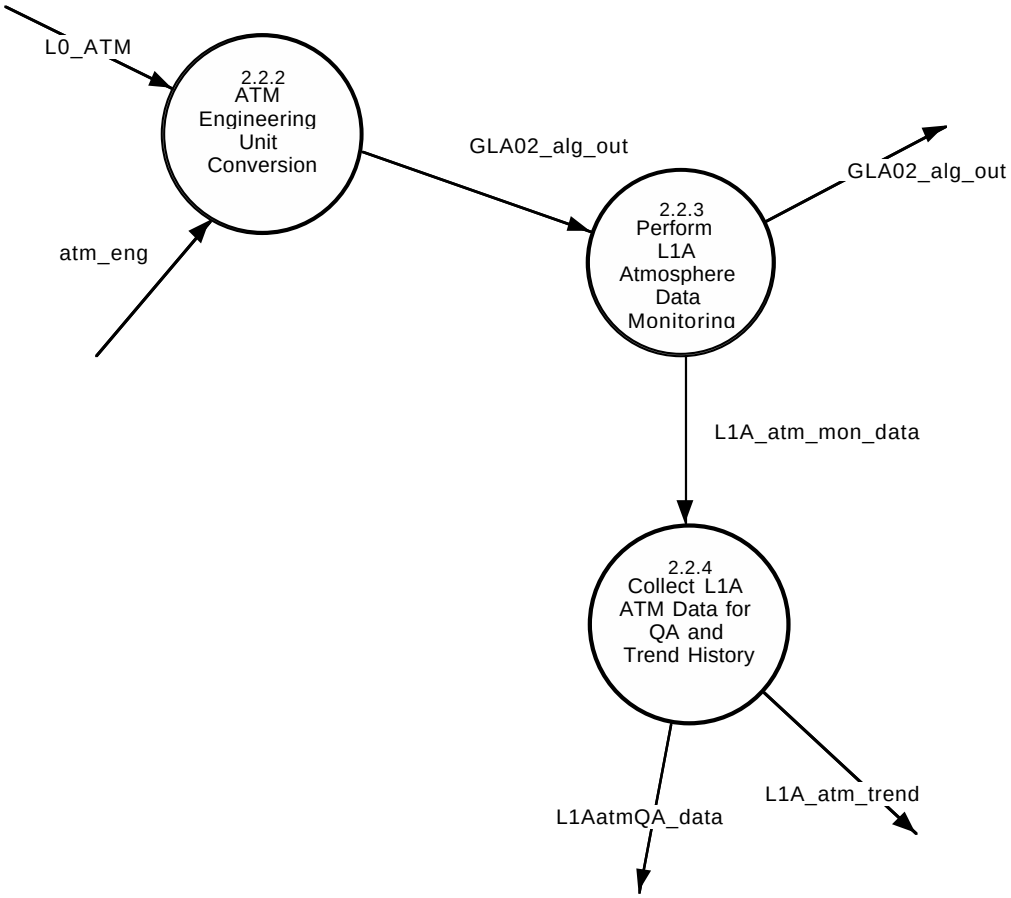

Figure 7-3 L1A LIDAR Processing (L_Atm)

are described in the following bullets.

- ATM Engineering Unit Conversion (L_atmeuc) - subprocess 2.2.2 performs engineering unit conversion on the raw Level 0 atmosphere data (L0_ATM) to

obtain the Level 1A atmosphere data in engineering units which are stored in the data structure GLA02_alg_out. Any engineering / housekeeping data that are required to be on the GLA02 data product are collected here and placed in the output structure.

- Perform ATM Data Monitoring (L_atmmon) - subprocess 2.2.3 compares the Level 1A atmosphere data (GLA02_alg_out) to preset monitoring limits and thresholds. Data required for QA and trend history (L1A_atm_mon_data) are output to subprocess 2.2.4. Flags are set appropriately in the L1A_atm_mon_data structure to indicate monitoring results.
- Collect ATM Data for QA and Trend History (L_atmqatrnd) - subprocess 2.2.4 takes the data generated during the 2.2.2 and 2.2.3 subprocesses and generates QA (L1AatmQA_data) and trend data (L1A_atm_trend).

7.3.3 Engineering Data Processing

The purpose of the Engineering Data Processing (process 2.3) is to generate the data for the Level 1A Engineering Data product (GLA03). The decomposition of process 2.3 is shown in Figure 7-4 "Engineering Data Processing (L_Eng)". This process deter-

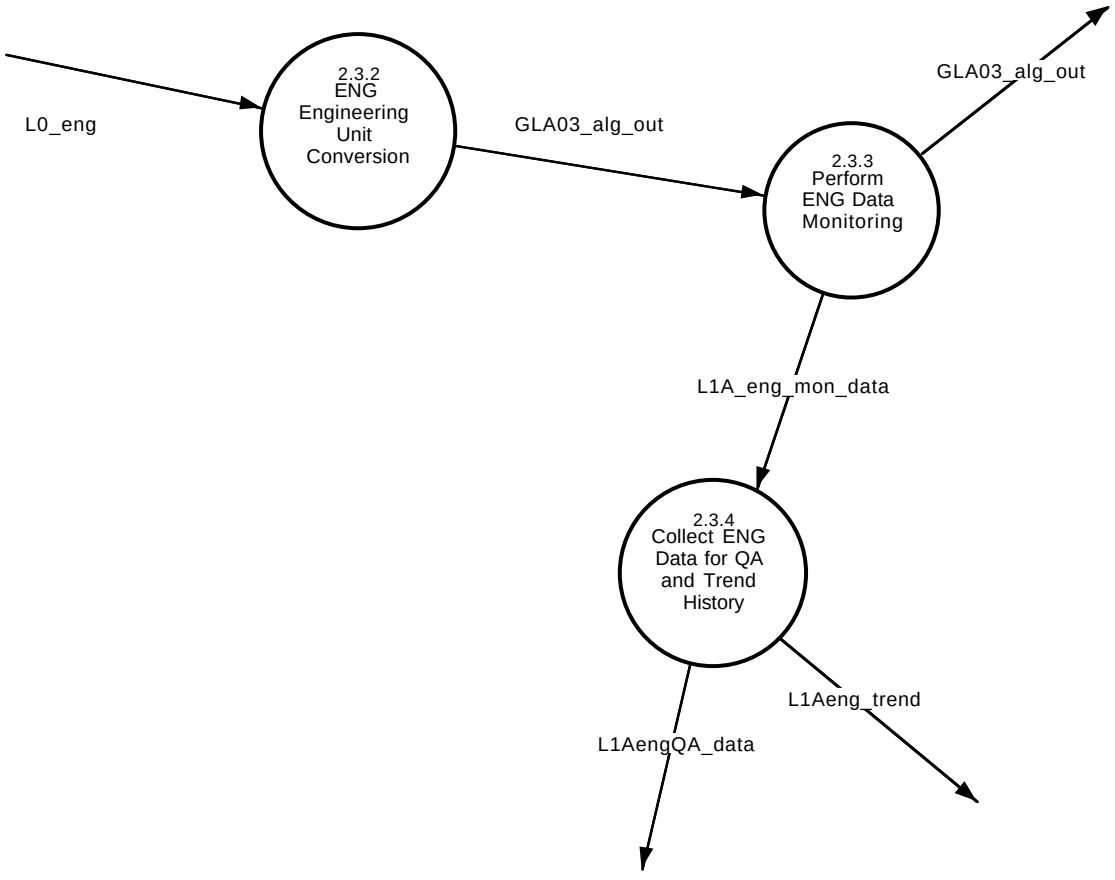

Figure 7-4 Engineering Data Processing (L_Eng)

mines the state of the instrument from the Level 0 data; converts the Level 0 engi-

neering data to engineering units; checks for data out of limits; and collects data for QA statistics and trend history. The subprocesses depicted in Figure 7-4 are described in the following bullets.

- ENG Engineering Unit Conversion (L_engeuc) - subprocess 2.3.2 performs engineering unit conversion on the raw Level 0 engineering data (L0_eng) to obtain the Level 1A engineering data which are stored in the output structure GLA03_alg_out. During this subprocess, the state of the instrument is determined, e.g., the operating laser and detector combination.
- Perform ENG Data Monitoring (L_engmon) - subprocess 2.3.3 compares the Level 1A engineering data (GLA03_alg_out) to preset monitoring limits and thresholds. This subprocess in particular evaluates the temperatures, currents, voltages, and status words to determine that the instrument is in good health. Data required for QA and trend history (L1A_eng_mon_data) are output to subprocess 2.3.4. Flags are set appropriately in L1A_eng_mon_data to indicate monitoring results.
- Collect ENG Data for QA and Trend History (L_engqatrnd) - subprocess 2.3.4 takes the data generated during the 2.3.2 and 2.3.3 subprocesses and generates QA (L1AengQA_data) and trend (L1Aeng_trend) data.

7.3.4 Collect Instrument and S/C Position and Attitude

The purpose of process 2.4 is to collect the GPS data, instrument and S/C position and attitude data and to generate the L1A Position and Attitude data product (GLA04). The GLA04 product is required for input to the precision orbit and attitude determination algorithms. The decomposition of process 2.4 is shown in Figure 7-5 "Collect Position and Attitude Data (L_Att)" on page 7-6. This process checks the Level 0 packets for errors, configures the data for output and collects QA and trend data. The subprocesses depicted in Figure 7-5 are described in the following bullets.

- Perform ATT Data Checks (L_attchkdata) - subprocess 2.4.1 checks the Level 0 Attitude packets (L0_POS_ATT) for completeness and for valid data. Any errors posatt_data_err) are sent to subprocess 2.4.3 for handling. During this subprocess any necessary flags are set indicating the state of the instrument. The Level 0 attitude data including the instrument state and any additional information (L0_POS_ATT_data) are sent to subprocess 2.4.2.
- Configure Data for Output (L_attconfig) - subprocess 2.4.2 reformats the data as required for the Level 1A Attitude Product file and is stored in the structure GLA04_alg_out. Data required for QA and trend history (PosAtt_QA) are output to subprocess 2.4.3.
- Collect ATT Data for QA and Trend History (L_attqatrnd) - subprocess 2.4.3 takes the data generated during the 2.4.1 and 2.4.2 subprocesses and generates QA (L1A_PosAttQA_data) and trend (L1A_PosAtt_trend) data.

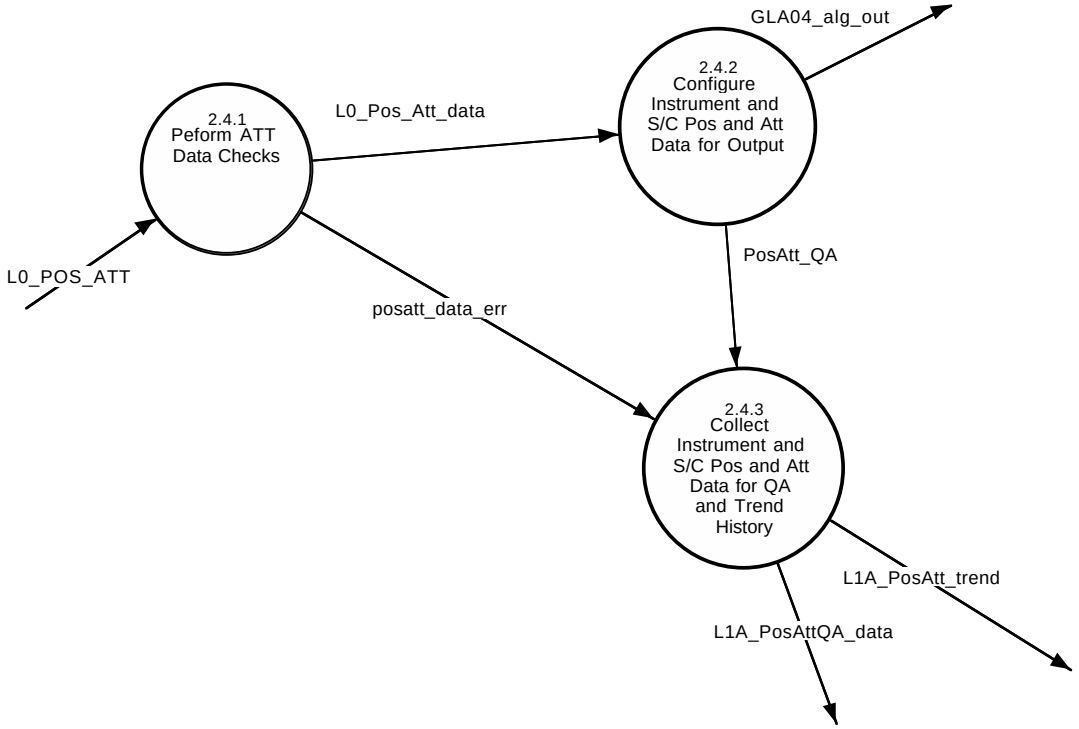

Figure 7-5 Collect Position and Attitude Data (L_Att)

7.3.5 Create L1A QA Statistics and Trend Data (L_QA_Trnd)

The purpose of process 2.5 is to compute the QA statistics from the data collected by the Level 1A Computations processes. This process also computes Level 1A trend data describing the performance of the instrument. The decomposition of process 2.5 is shown in Figure 7-6 "Collect L1A QA Statistics and Trend Data" on page 7-7. The subprocesses depicted in Figure 7-6 are described in the following bullets.

- Compute L1A QA and Processing Stats (L_compstat) - subprocess 2.5.1 uses the QA data (L1AaltQA_data, L1AatmQA_data, L1AengQA_data, L1APosAttQA_data, predLocQA) generated by the L1A Computations processes to generate the cumulative QA statistics (L1AQA) describing the Level 1A data products and their generation.
- Compute Trend History Data (L_comptrnd) - subprocess 2.5.2 uses the trend data (L1A_alt_trend, L1A_PosAtt_trend, L1A_lid_trend, L1A_eng_trend) generated by the L1A Computations processes to generate the trend data (L1Atrend_data) describing the Level 1A data products and the performance and health of the instrument.

7.3.6 Get Predicted Location

The Get Predicted Location process (process 2.6) obtains the latitude and longitude for each shot from the predicted orbit file using common routines. The shot times are input, the predicted orbit file is interpolated to get the spacecraft position vector for

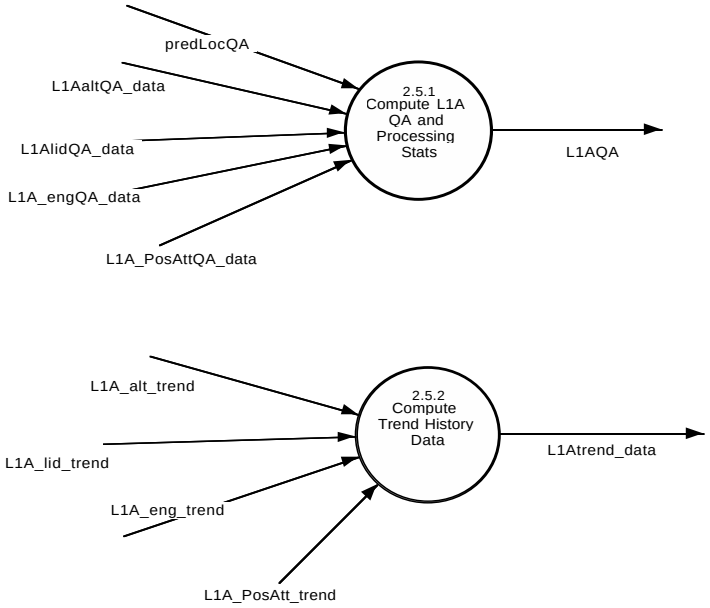

Figure 7-6 Collect L1A QA Statistics and Trend Data

each shot time, and then the latitude and longitude are computed from the position vector. The common routines c_intrpPOD and c_calc_sploc will be used for the calculations. The common routines will be called directly by the L1A Manager; it is not necessary to generate code for the Level 1A Computations subsystem.

7.3.7 Calculate Shot Time

The Calculate Shot Time process (process 2.7) will generate the precise time of each laser shot. The time will be calculated from the laser fire time, GPS time, and the GPS latch time. Offsets and calibration factors will applied as necessary.

Section 8

Level 1B Waveform Processes

8.1 Function

The Level 1B Waveforms subsystem computes the geolocation, and produces waveform-based information required to produce the elevation products (GLA05_SCF).

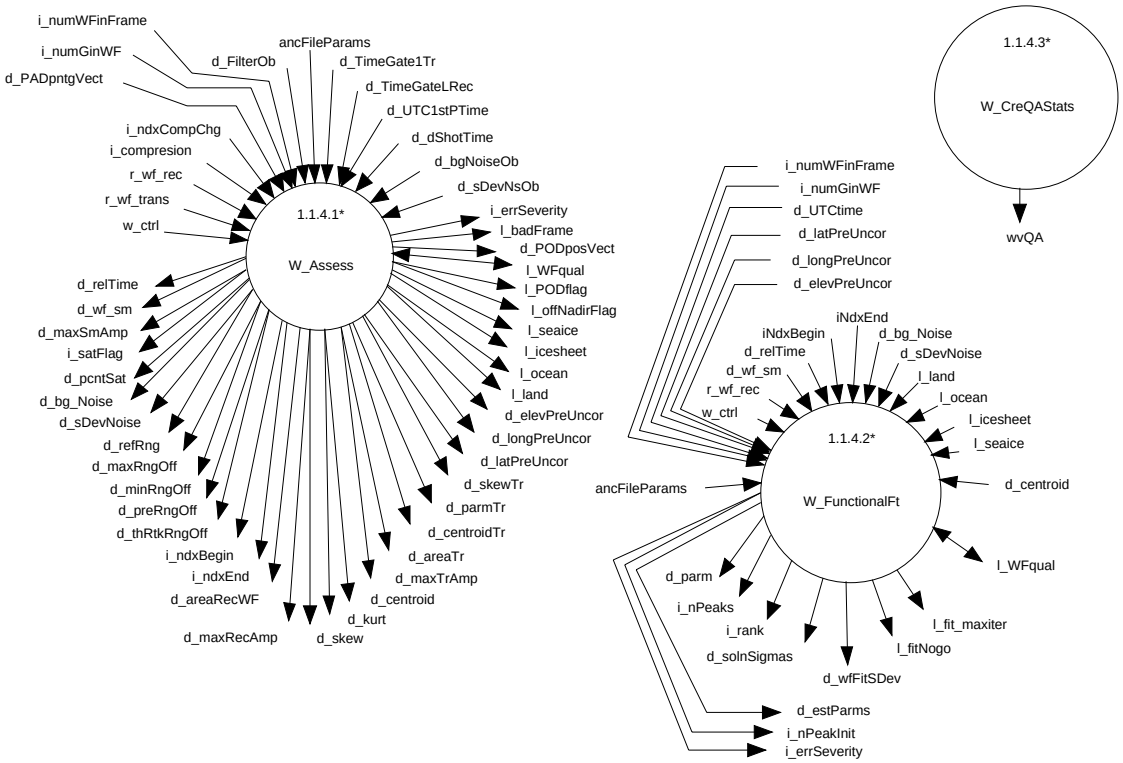

Figure 8-1 Level 1B Waveforms

8.2 Design Constraints / Decisions / Assumptions

Data will be passed to the waveform processing manager one1-second frame at a time. All data will be passed to the lowest level possible as arrays in order to take advantage of Fortran 90's matrix arithmetic.

8.3 DFDs and their Descriptions

The Level 1B Waveforms subsystem is divided into three processes (W_Assess, W_FunctionalFt, & W_CreQAStats) which generate waveform-based information required to produce the elevation products (GLA05). A control flag (w_ctrl) is passed to processes W_Assess and W_FunctionalFt indicating whether processing will be

land algorithm only, other-than-land algorithm only, or both, and whether the subprocess W_DetGeo will be called. In addition to producing waveform-based information, processes W_Assess and W_FunctionalFt generate QA data which is retrieved by W_CreQAStats, where QA statistics are created for the subsystem (wvQA) for inclusion in the summary information product (ANC06). The following is a description of each of the processes.

8.3.1 Assess Waveforms & Calc Std Range Offset (W_Assess)

Utilizes the transmitted waveforms (r_wf_trans), received waveforms (r_wf_rec), compression ratios (i_compression), index of compression change (i_ndxCompChg), the PAD pointing vector (d_PADpntgVect), the number of gates in each waveform (i_numGinWF, normally 200 or 544), the number of waveforms in a frame (i_numWFinFrame, normally 40), the filter used by the spacecraft (d_filterOb), the UTC time of the first waveform (d_UTC1stPTime), the time increment from d_UTC1stPTime to each of the 39 other waveforms in the frame (d_dShotTime), the digitizer time of gate 1 of the transmitted pulse (d_TimeGate1Tr), the digitizer time of the last gate of the received waveform (d_TimeGateLRec), background noise (d_bgNoise), the standard deviation of the background noise level calculated by the on board algorithm (d_sDevNsOb - snoise_ob in ATBD), and an array of flags (l_WFqual - indicates if any of the waveforms are invalid) to perform a general assessment of the waveforms including: a check for waveform saturation (i_satFlag, d_pcntSat); defining the noise and the begin and end of signal (d_bg_Noise, d_sDevNoise, i_ndxBegin, d_minRngOff, i_ndxEnd, d_maxRngOff); characterizing the transmitted pulse (d_maxTrAmp, d_areaTr, d_centroidTr, d_parmTr, d_skewTr); computing smoothed waveforms (d_wf_sm, d_maxSmAmp); calculating various shape characteristics for the received waveform (d_maxRecAmp, d_skew, d_kurt, d_centroid); calculating the reference range (d_refRng), and range offsets (d_preRngOff, d_thRtkRngOff); and determining the preliminary latitude, longitude and elevation (d_latPreUncor, d_longPreUncor, d_elevPreUncor).

d_relTime is the relative time of each gate taking account of the compression information. For example, if the original gates were one ns apart, and there was a compression ratio of 2, then the time of gate 1 (r_wf_rec[1]) would be the average of the times of the first two digitizer gates ((0+1)/2 or 0.5).

d_wf_sm is the smoothed waveform, and d_maxSmAmp is the maximum amplitude of the smoothed waveform.

i_sat_flag indicates whether waveform signal saturation is minimal (normal waveform processing is applied), moderate (some information on elevation can be approximated) or excessive (no waveform processing is applied).

d_pcntSat is the percent saturation for each waveform.

d_bg_Noise is either the observed noise (d_bgNoiseOb), or the calculated background noise, and d_sDevNoise is either the observed noise standard deviation (d_sDevNsOb), or the standard deviation of the calculated noise. The calculation is

performed if anc07%i_nsCal is set, or if the observed noise is zero or invalid and the waveform is not invalid.

d_refRng is the reference range in ns (the time from the centroid of the transmitted pulse to the time of last gate of the received waveform).

d_maxRngOff is the offset to be added to d_refRng to give the time of the last threshold crossing (farthest from the spacecraft).

d_minRngOff is the offset to be added to d_refRng to give the time of the first threshold crossing (closest to the spacecraft).

d_preRngOff is the offset to be added to d_refRng to give the time of the preliminary uncorrected range (threshold farthest from spacecraft).

d_thRtkRngOff is the offset to be added to d_refRng to give the time of the retracker threshold.

i_ndxBegin and i_ndxEnd are the indices of the beginning and end of signal of the received waveform.

d_areaRecWF is the area under the received waveform, above the noise, from signal begin through signal end.

d_maxRecAmp is the maximum amplitude of the received waveform.

d_skew is the skewness of the received waveform from signal begin through signal end.

d_kurt is the kurtosis of the received waveform from signal begin through signal end.

d_centroid is the centroid of the received waveform from signal begin through signal end.

d_maxTrAmp is the maximum amplitude of the transmitted pulse.

d_areaTr is the area under the transmitted pulse (all 32 gates).

d_centroidTr is the centroid of the transmitted pulse (all 32 gates).

d_parmTr is the gaussian amplitude, peak location relative to gate one of the transmitted pulse, and gaussian sigma of the single peak gaussian fit to the transmitted pulse.

d_skewTr is the skewness of the transmitted pulse.

d_latPreUncor, d_longPreUncor, and d_elevPreUncor are the preliminary, uncorrected latitude, longitude, and elevation.

l_land, l_ocean, l_icesheet, and l_seaice are flags taken from the region mask indicating the possible presence of land, ocean, icesheet and / or seaice for each waveform.

l_offNadirFlag indicates if the spacecraft is pointed off nadir.

l_PODflag indicates the status of the orbit information.

l_WFqual is an array of 31 flags for each waveform. These flags include: no signal, no leading edge, no trailing edge, no transmitted pulse, land, ocean, icesheet, seaice, no fit, noise and standard deviation of noise are calculated, maximum iterations during fit, region selected for waveform fit, and invalid waveform.

d_PODposVect is the precision orbit position vector.

l_badFrame indicates if the entire frame is bad.

i_errSeverity indicates the maximum severity of any error occurring during the execution of W_Assess.

8.3.2 Calculate the WF Functional Fit (W_FunctionalFt)

Utilizes the waveform (r_wf_rec), the smoothed waveform (d_wf_sm), time (d_relTime), time array indices for the begin & end of signal (iNdxBegin & iNdxEnd), the background noise (d_bg_Noise), the standard deviation of the background noise (d_sDevNoise), the centroid of the received waveform (d_centroid), and the region type(s) (l_land, l_ocean, l_icesheet, and l_seaice) to fit the waveform to a function and return the calculated waveform function parameters (d_parm), the estimated parameters (d_estParms), the initial number of peaks found (i_nPeakInit), the number of solution peaks found (i_nPeaks), the ranks of the peaks (i_rank), the solution sigmas (d_solnSigmas), the standard deviation of the fit (d_wfFitSDev), a flag indicating if the fit was unsuccessful - i.e. if the normal matrix turns singular (l_fitNogo), and a flag indicating if the fit ended without meeting the convergence criteria (l_fit_maxiter).

8.3.3 Create Waveforms QA Statistics (W_CreQAStats)

Calls W_GetAsQAStats, and W_GetFfQAStats to retrieve QA data, combines that data and returns it to the waveform submanager (wvQA).

wvQA includes for ice sheet and land products: the percent of measurements for which no signal could be found; for measurements where a signal was found - the percent of measurements that could not be processed due to saturation; and histograms of the skewness of each single pulse return and the percent of saturated signal compared to real signal within each waveform from sig_beg to sig_end, for all products: the goodness of fit of the Gaussian to the waveforms; for sea-ice, ice sheet and land products: the number of peaks in the smoothed waveforms; for ice sheet and land products for measurements where a signal could be found: the percent of measurements for which the fitting procedure did not converge, a histogram of the differences between the centroid of the smoothed waveform and the centroid of the Gaussian fit to the last peak, and a histogram of the standard deviation of the fit to the raw waveform; for sea-ice products: histograms of the time delays expressed as distances for centroid to center peak, sig_beg to center peak, sig_end to center peak; for ocean products: histograms of time delays expressed as distances for centroid to sig_beg, and centroid to sig_end.

The Assess Waveforms process is divided into eleven subprocesses (W_CalcRelTime, W_CharTrPulse, W_CalcNoise, W_CalcRefRng, W_SmoothPreRC, W_DetGeo,

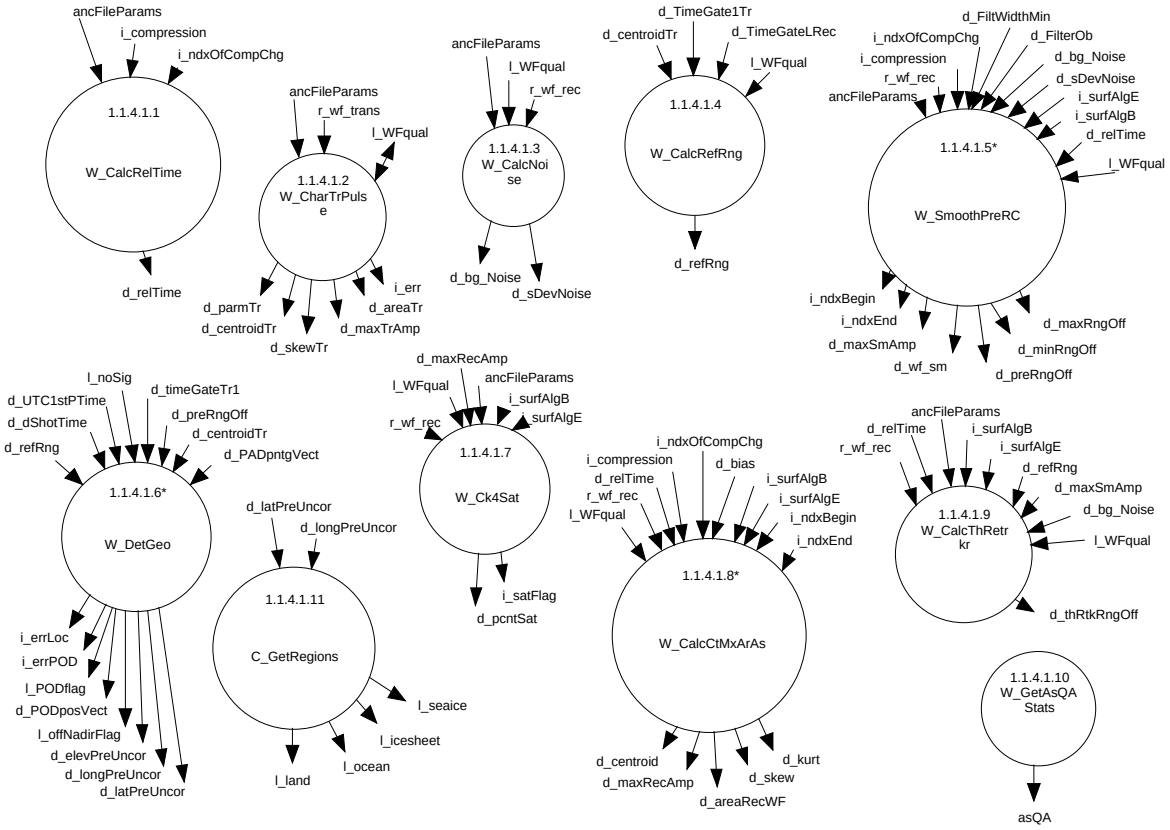

Figure 8-2 Assess Waveforms & Calculate Standard Range Corrector

C_GetRegions, W_Ck4Sat, W_CalcCtMxArAs, W_CalcThRetrkr, and W_GetAsQAStats) which perform a general assessment of the waveforms. In addition to assessing the waveforms, the subprocesses generate QA data (stored in the W_Assess_mod module) which is retrieved by subprocess W_GetAsQAStats. Following is a description of each subprocess:

8.3.4 Calculate Relative Time (W_CalcRelTime)

Utilizes the compression ratios (i_compression), and the index of the change of compression (i_ndxOfCompChg) to determine the relative time (d_relTime).

8.3.5 Characterize Transmitted Pulse (W_CharTrPulse)

Utilizes the transmitted pulse (r_wf_trans), and the waveform quality flags (l_WFqual) to determine the transmitted pulse characteristics: gaussian fit parameters (d_parmTr), centroid (d_centroidTr), skewness (d_skewTr), maximum amplitude (d_maxTrAmp), and area under the pulse (d_areaTr).

8.3.6 Calculate Noise (W_CalcNoise)

Utilizes the received waveform (r_wf_rec), and the waveform quality flags (l_WFqual) to determine the background noise and the standard deviation of the background noise (d_bg_Noise, d_sDevNoise).

8.3.7 Calculate Reference Range Offset (W_CalcRefRng)

Utilizes the centroid of the transmitted pulse (d_centroidTr), the digitizer time of the first gate of the transmitted pulse (d_TimeGate1Tr), the digitizer time of the last gate of the received waveform (d_TimeGateLRec), and the waveform quality flags (l_WFqual) to calculate the reference range offset (d_refRng).

8.3.8 Calculate Smooth Waveform and Range Offset (W_SmoothPreRC)

Utilizes the received waveform (r_wf_rec), the compression ratios (i_compression), the index of change of compression (i_ndxOfCompChg), the minimum filter width (d_FiltWidthMin), the observed filter width (d_FilterOb), the background noise (d_bg_Noise), the standard deviation of the noise (d_sDevNoise), the relative time (d_relTime), which set(s) of ancillary parameters to use (i_surfAlgB, i_surfAlgE), and the waveform quality flags (l_WFqual), and the subroutine W_Smooth1, to calculate the smoothed waveform (d_wf_sm), the maximum amplitude of the smoothed waveform (d_maxSmAmp), signal begin (i_ndxBegin), signal end (i_ndx_End), the maximum range offset (d_maxRngOff), the minimum range offset (d_minRngOff), and the preliminary uncorrected range offset (d_preRngOff).

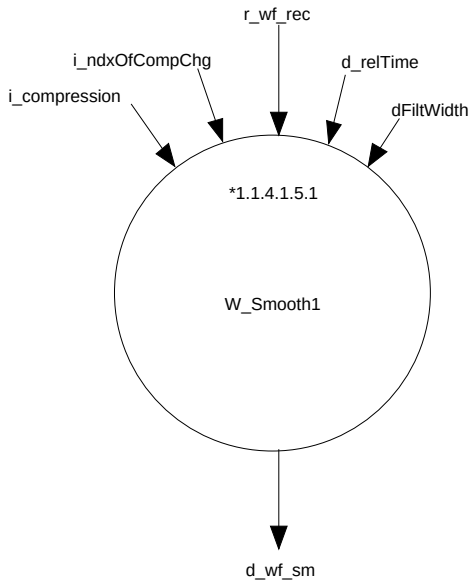

Figure 8-3 Calculate Smoothed Waveform

8.3.9 Determine Geolocation (W_DetGeo)

Utilizes the reference range (d_refRng), the UTC time of the first waveform in the frame (d_UTC1stPTime), the time relative to the first waveform for the other wave-

forms in the frame (d_dShotTime), the no-signal flag (l_noSig), the digitizer time of the first gate of the transmitted pulse (d_timeGateTr1), the preliminary range offset (d_preRngOff), the centroid of the transmitted pulse (d_centroidTr), and the attitude vector (d_PADpntgVect), and the subroutines C_IntrpPOD, and C_CalcSpLoc to calculate the preliminary, uncorrected latitude, longitude, and elevation (d_latPreUncor, d_longPreUncor, d_elevPreUncor), and returns the orbital position vector (d_PODposVect), and flags indicating an off-nadir pointing (l_offNadirFlag), POD status (l_PODflag), and error status (i_errLoc, i_errPOD).

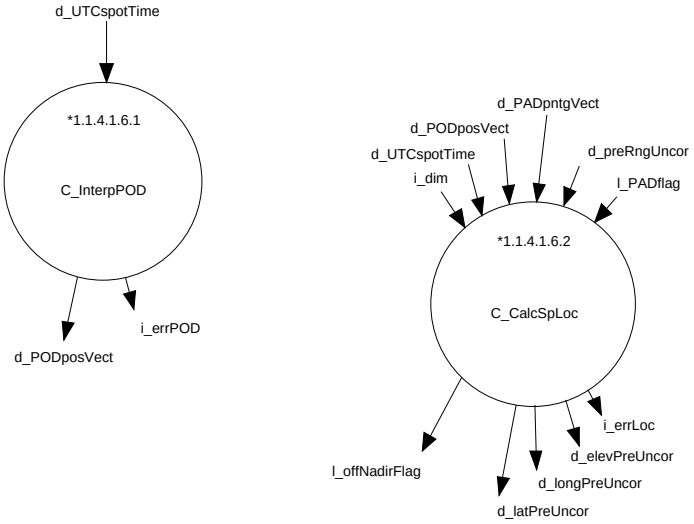

Figure 8-4 Determine Geolocation

8.3.10 Check for Saturation (W_Ck4Sat)

Utilizes the received waveform (r_wf_rec) and waveform signal characteristics (d_maxRecAmp, l_WFqual, i_surfAlgB, i_surfAlgE) to calculate the percent saturation (d_pcntSat), and determine if the waveform should be processed (i_satFlag).

8.3.11 Calculate Centroid, Max, Area, and Asymmetry (W_CtMxArAs)

Utilizes the received waveform (r_wf_rec) and waveform signal characteristics (l_WFqual, d_relTime, i_compression, i_ndxOfCompChg, d_bias, i_surfAlgB, i_surfAlgE, i_ndxBegin, i_ndxEnd), and the subroutines W_CalcArea, W_CalcCent, and W_CalcSkewKurt, to calculate the centroid, maximum received amplitude, area, skewness, and kurtosis of the waveform (d_centroid, d_maxRecAmp, d_areaRecWF, d_skew, d_kurt).

8.3.12 Calculate Threshold Retracker Offset (W_CalcThRetrkr)

Utilizes the received waveform (r_wf_rec), and waveform signal characteristics (d_relTime, i_surfAlgB, i_surfAlgE, d_refRng, d_maxSmAmp, d_bg_Noise, l_WFqual) to calculate the threshold retracker range offset (d_thRtkRngOff).

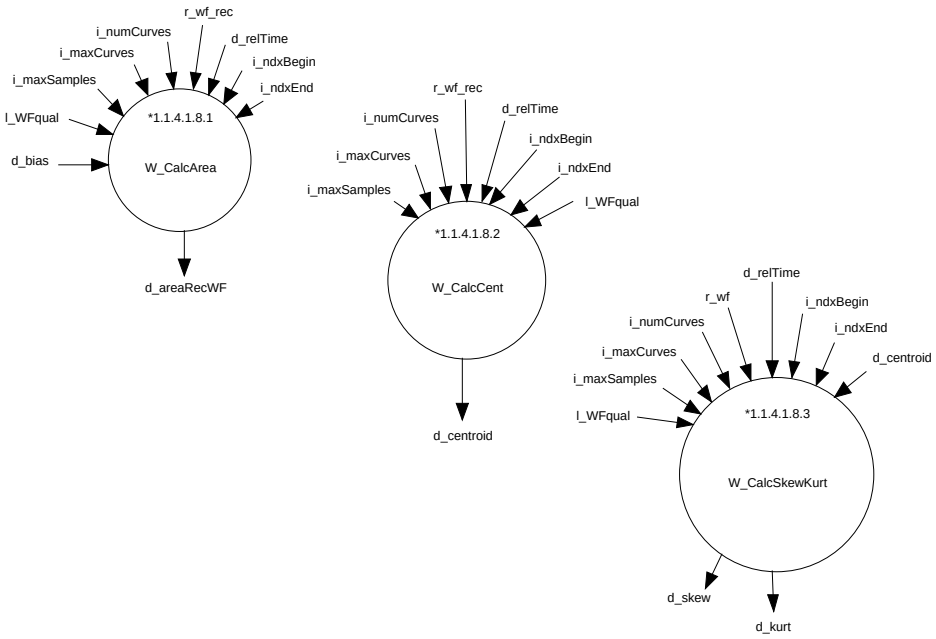

Figure 8-5 Calculate Centroid, Max, Area, Asymmetry

8.3.13 Get W_Assess QA Statistics (W_GetAsQAStats)

Retrieves the QA statistics (as QA) accumulated by W_Assess and its subprocesses.

8.3.14 Get Regions (C_GetRegions)

Utilizes the preliminary uncorrected latitude and longitude (d_latPreUncor, d_longPreUncor) to determine the region type(s) for each waveform (l_land, l_ocean, l_icesheet, l_seaice).

8.3.15 Smooth One Waveform (W_Smooth1)

Utilizes the received waveform (r_wf_rec), the compression ratios (i_compression), the index of compression change (i_ndxOfCompChg), the relative time (d_relTime), and the filter width (dFiltWidth) to calculate the smoothed waveform (d_wf_sm).

The Determine Geolocation process is divided into two subprocesses (C_IntrpPOD, and C_CalcSpLoc) which determine the geolocation of the spot on the earth where the transmitted pulse is reflected. Following is a description of each subprocess:

8.3.16 Interpolate Orbit (C_IntrpPOD)

Utilizes the time for the transmitted pulse to return to the instrument (d_time) and precision orbit data (use POD_FileInfo to access) to determine the POD fixed position vector (POD_pos_vect), and POD flag (l_PODflag).

8.3.17 Calculate Elevation and Geolocation (C_CalcSpLoc)

Utilizes precision orbit data (POD_pos_vect), precision attitude data (PAD_pointngVect), time (d_UTCspotTime), and the standard range (d_preRngUncor) to determine the uncorrected latitude, longitude, and standard elevation (d_latPreUncor, d_longPreUncor, d_elevPreUncor), the off nadir flag (l_offNadirFlag), and the off nadir angle (d_offNdrAngD).

8.3.18 Calculate Area (W_CalcArea)

Utilizes the received waveform (r_wf_rec), the relative time (d_relTime), signal begin and end (i_ndxBegin, i_ndxEnd), the waveform quality flags (l_WFqual), the number of gates in each waveform (i_maxSamples), the maximum number of waveforms in the frame (i_maxCurves), the number of waveforms in the frame (i_numCurves), and the bias (d_bias) to calculate the area under the received waveform from signal begin to signal end (d_areaRecWF).

8.3.19 Calculate Centroid (W_CalcCent)

Utilizes the received waveform (r_wf_rec), the relative time (d_relTime), signal begin and end (i_ndxBegin, i_ndxEnd), the waveform quality flags (l_WFqual), the number of gates in each waveform (i_maxSamples), the maximum number of waveforms in the frame (i_maxCurves), and the number of waveforms in the frame (i_numCurves) to calculate the centroid of the received waveform from signal begin to signal end (d_centroid).

8.3.20 Calculate Skewness and Kurtosis (W_CalcSkewKurt)

Utilizes the received waveform (r_wf), the relative time (d_relTime), signal begin and end (i_ndxBegin, i_ndxEnd), the waveform quality flags (l_WFqual), the number of gates in each waveform (i_maxSamples), the maximum number of waveforms in the frame (i_maxCurves), the number of waveforms in the frame (i_numCurves), and the centroid (d_centroid) to calculate the skewness and kurtosis of the received waveform from signal begin to signal end (d_skew, d_kurt).

The Calculate Other WF Characteristics process consists of one subprocess (W_ParamWithFit) which calculates other waveform parameters. The following is a description of the subprocess.

8.3.21 Parameterize WF with Fit (W_ParamWithFit)

Utilizes the waveform (r_wf_rec), the smoothed waveform (d_wf_sm), time (d_relTime), time array indices for the begin & end of signal (i_ndxBegin & i_ndxEnd), the background noise (d_bg_Noise), the standard deviation of the noise (d_sDevNoise), expected surface types (l_land, l_ocean, l_icesheet, l_seaice), the maximum number of iterations allowed (i_maxiter), a value used to determine the minimum amplitude for a peak (d_nPeak_min), the centroid (d_centroid), the waveform quality flags (l_WFqual), and the maximum number of peaks allowed in a fitted waveform (i_maxfit) to fit the waveform to a function and return the calculated waveform function parameters (d_params), the initially estimated parameters (d_estParams), the number of peaks found (i_nPeaks), the number of peaks found

Calculate Other WF Characteristics Subprocesses

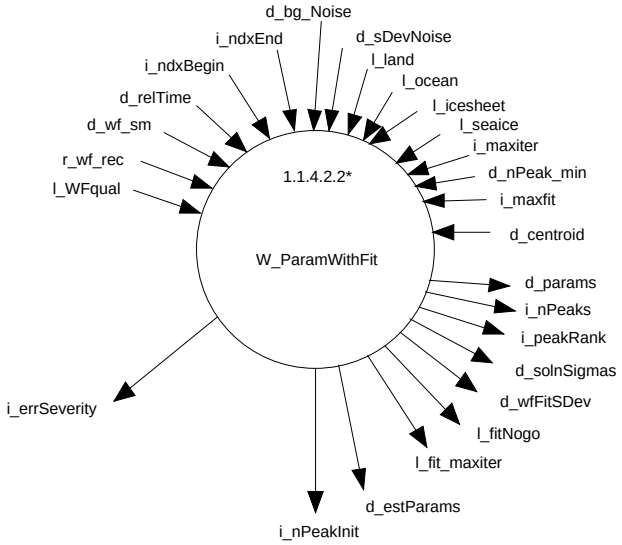

Figure 8-6 Calculate Other Waveform Characteristics

before the fitting process (i_nPeaksInit), the ranks of the solution peaks (i_peakRank), the solution sigmas (d_solnSigmas), the standard deviation of the fit (d_wfFitSDev), a flag indicating if the fit was unsuccessful - i.e. if the normal matrix turns singular (l_fitNogo), a flag indicating if the fit ended without meeting the convergence criteria (l_fit_maxiter), and a status flag (i_errSeverity).

Section 9

Level 1B and 2 Atmosphere Processes

9.1 Function

The function of the Levels 1B and 2 Atmosphere Computations subsystem is to create atmosphere parameters for the standard data products and to generate associated metadata and quality assessment (QA) data.

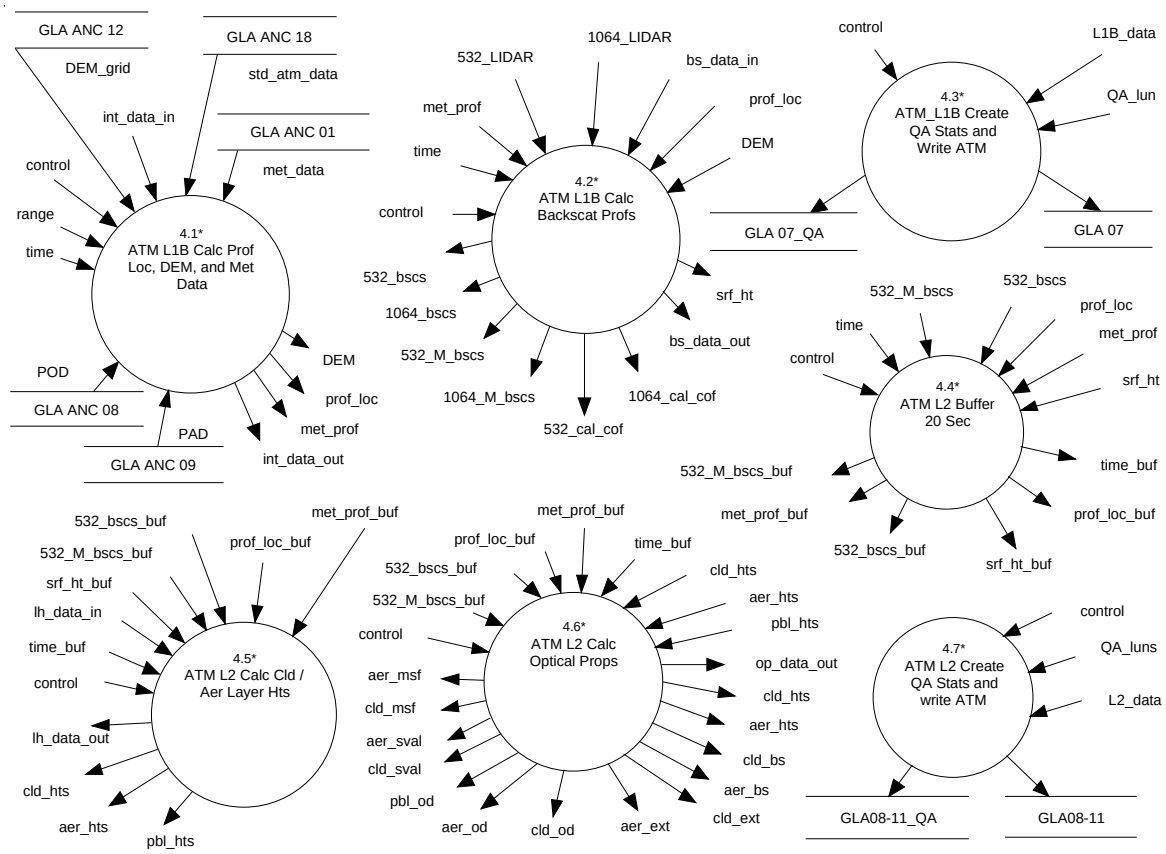

Figure 9-1 Atmosphere Computations

9.2 Design Constraints / Decisions / Assumptions

- Data will be passed to the atmosphere processing manager one 1-second record at a time.
- The Level 1B product (GLA07) is output at one record per 1 sec.
- The processing of Level 2 data is buffered for 20 sec irrespective of time gaps between data records.
- The Level 2 products (GLA08-11) are output at one record per 4 sec.

- Cloud products are reported at once per 4 sec, 1 sec, and 5 Hz from 20 to 0 km, and at 40 Hz below 4 km.
- Aerosol products are reported at once per 4 sec from 20 to 0 km and at once per 20 sec from 40 to 20 km.
- Twenty second averaging requires that at least ten seconds of valid profiles are available. Likewise, four second averaging requires that at least two seconds of valid profiles are available.

Met data sets at times before and after the time of the profile are interpolated to the time of the profile. If either of the met data sets are missing, then the available met data set is used without interpolation. If no met data sets are available, then standard atmosphere data are used instead.

9.3 DFDs and their Descriptions

9.3.1 Calculate Profile Locations and Met Data

The function of the ATM L1B Calculate Profile Locations and Met Data process is to create parameters for the Level 1B calibrated backscatter product GLA07. Records are written to this product once per second. This process geolocates the data and creates meteorological arrays. It utilizes data from ancillary files GLA ANC 01: meteorological (met) data and GLA ANC 18: standard atmosphere (std atm) data. It also uses the precision orbit determination (POD) position vector, time, and range (from GLA02).

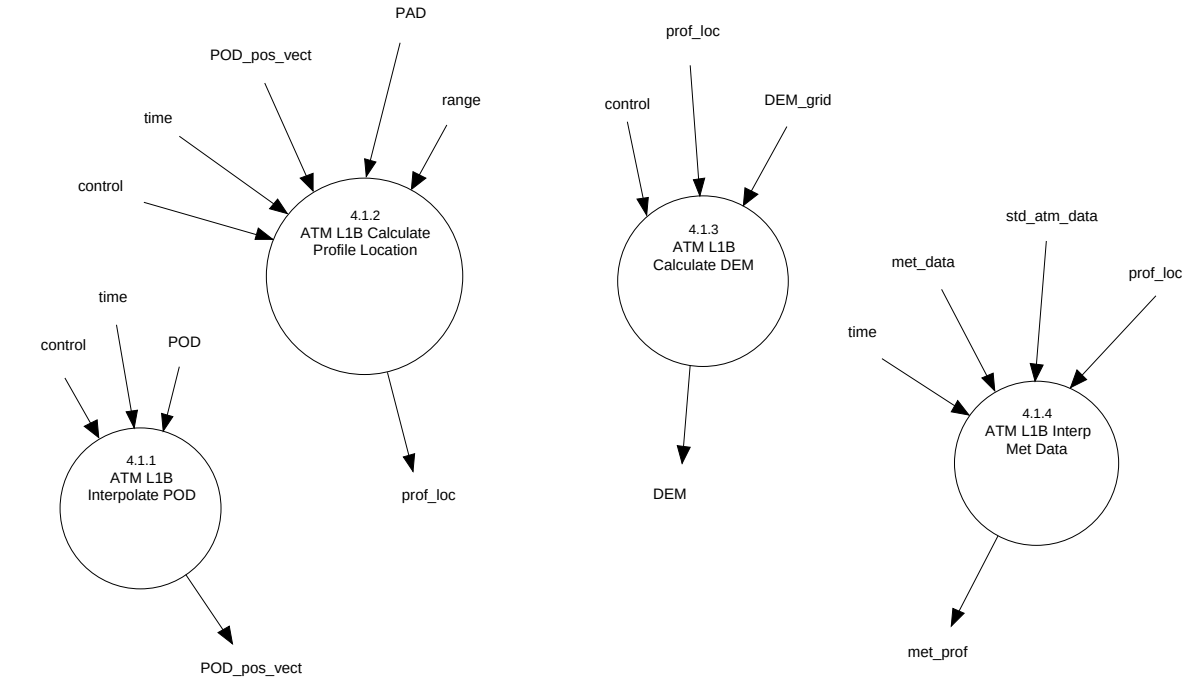

Figure 9-2 Atmosphere Subsystem: Profile Location / Met

9.3.2 Calculate Backscatter Cross Section Profiles

The function of the ATM L1B Calculate Backscatter Cross Section Profiles process is to create parameters for the Level 1B calibrated backscatter product GLA07. Records are written to this product once per second. This process creates the attenuated backscatter cross section profiles. It utilizes data from ancillary file GLA ANC 12: digital elevation model (DEM). It also uses meteorological data, time, and lidar output from GLA02.

9.3.3 Calculate Layer Heights

The function of the ATM L2 Calculate Layer Heights process is to create parameters for the Level 2 aerosol layer height product GLA08 and the Level 2 cloud layer height product GLA09. Records are written to these products once per four seconds. This process determines, at several resolutions, the top and bottom elevations of multiple aerosol and cloud layers, ground detections, and the planetary boundary layer (PBL) height. It utilizes meteorological data, time, and the lidar and molecular backscatter profiles from GLA07.

9.3.4 Calculate Optical Properties

The function of the ATM L2 Calculate Optical Properties process is to create parameters for the Level 2 cloud and aerosol backscatter cross section product GLA10 and the Level 2 optical depth product GLA11. Records are written to these products once per four seconds. This process creates cross section profiles for cloud and aerosol backscatter and extinction as well as optical depths for the cloud and aerosol layers. It utilizes meteorological data, time, and the lidar and molecular backscatter profiles from GLA07. It also uses the cloud layer heights and ground detections from GLA09 and the elevated aerosol and PBL layer heights from GLA08.

In addition to producing product parameters, the processes generate QA data which enter the ATM L1B/L2 Create QA Statistics process, where QA statistics are compiled for the subsystem for inclusion in the summary information product GLA ANC 06.

The ATM L1B Calculate Profile Locations and Met Data process is divided into two subprocesses. Following is a description of each subprocess:

9.3.5 ATM L1B Calculate Profile Locations (C_CalcSpLoc)

Utilizes the POD position vector (POD_pos_vect), time, and range (atm_range) to generate the profile location (prof_loc) at 1 second. This is a common routine used by several processes. This process is under control which means that it is only selectively called by the atmosphere manager based upon the processing scenario selected in an input control file.

9.3.6 ATM L1B Interpolate Met Data (A_interp_met)

Interpolates and combines met data from GLA ANC 01 and std atm data from GLA ANC 18 to generate met/std atm profiles at 1 second (met_prof) using profile location (prof_loc) and time.

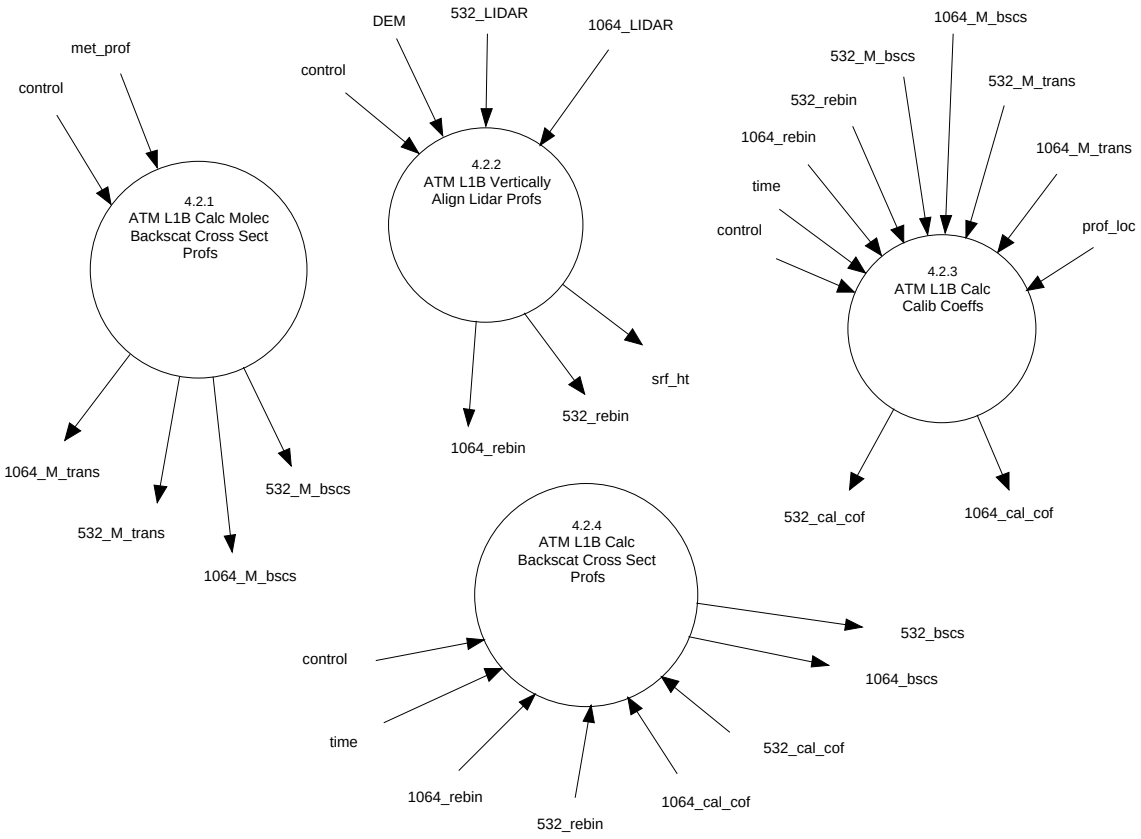

Figure 9-3 Atmosphere Subsystem: Backscatter Subprocesses

The ATM L1B Calculate Backscatter Cross Section Profiles process is divided into four subprocesses. The following is a description of each subprocess.

9.3.7 ATM L1B Calculate Molecular Backscatter Cross Sections (A_mbscs)

Utilizes met/std atm profiles at 1 second (met_prof) to create molecular backscatter cross section profiles at 532 nm (532_M_bscs) and 1064 nm (1064_M_bscs) at 1 second. Also created are molecular transmission profiles at 532 nm (532_M_trans) and 1064 nm (1064_M_trans) at 1 second. This process is under control which means that it is only selectively called by the atmosphere manager based upon the processing scenario selected in an input control file.

9.3.8 ATM L1B Vertically Align Profiles (A_rebin_lid)

Combines and vertically aligns 532 nm lidar signals at 1, 5 and 40 Hz (532_LIDAR) and 1064 nm lidar signals at 5 and 40 Hz (1064_LIDAR) creating 532 nm profiles (532_rebin) and 1064 nm profiles (1064_rebin) at 5 and 40 Hz. Utilizes DEM from GLA ANC 12 at 1 sec, but if the DEM is corrupt, uses last valid DEM, so the DEM used (atm_dem) is also output.

9.3.9 ATM L1B Calculate Calibration Coefficients (A_cal_cofs)

Utilizes molecular backscatter section profiles at 532 nm (532_M_bscs) and 1064 nm (1064_M_bscs), molecular transmission profiles at 532 nm (532_M_trans) and 1064 nm (1064_M_trans), background lidar signal at 532 nm (532_bkgd), profile location (prof_loc), time, and vertically aligned lidar profiles at 532 nm (532_rebin) and at 1064 nm (1064_rebin) to create backscatter calibration coefficients at 532 nm (532_cal_cof) and 1064 nm (1064_cal_cof) at 1 sec.

9.3.10 ATM L1B Calculate Backscatter Cross Section Profiles (A_ bscs)

Utilizes vertically aligned lidar profiles at 532 nm (532_rebin) and 1064 nm (1064_rebin), backscatter calibration coefficients at 532 nm (532_cal_cof) and 1064 nm (1064_cal_cof), and time to create attenuated backscatter cross section profiles at 532 nm at 5 and 40 Hz (532_bscs) and at 1064 nm at 5 and 40 Hz (1064_bscs). If the lidar signal at 532 nm is saturated, then the 532 nm backscatter value is replaced with the corresponding 1064 nm backscatter value.

Note: It will also be an option to either use the calibration coefficients calculated in subprocess A_cal_cofs or to use lab-measured coefficients instead, since the calculated coefficients, especially the one at 1064 nm, may be unreliable due to low signal at high altitude.

The ATM L2 Calculate Cloud / Aerosol Layer Heights process is divided into three subprocesses. The following is a description of each subprocess.

9.3.11 ATM L2 Calculate Cloud Layers (A_cld_lays)

Utilizes attenuated backscatter cross section profiles at 532 nm at 5 Hz and 40 Hz (532_bscs), molecular backscatter cross section profiles at 532 nm (532_M_bscs), DEM (atm_dem), and time to create cloud layer heights and ground detections at 4 sec and 1, 5 and 40 Hz (cld_hts). This process is under control which means that it is only selectively called by the atmosphere manager based upon the processing scenario selected in an input control file.

9.3.12 ATM L2 Calculate PBL Layer (A_pbl_lay)

Utilizes attenuated backscatter cross section profiles at 532 nm at 5 Hz (532_bscs), cloud layer heights at 4 sec and 5 Hz (cld_hts), DEM (atm_dem), and time to create PBL height and ground detection at 4 sec and 5 Hz (pbl_ht). This process is under control which means that it is only selectively called by the atmosphere manager based upon the processing scenario selected in an input control file.

9.3.13 ATM L2 Calculate Elevated Aerosol Layers (A_aer_lays)

Utilizes attenuated backscatter cross section profiles at 532 nm at 5 Hz (532_bscs), molecular backscatter cross section profiles at 532 nm (532_M_bscs), met / std atm profiles (met_prof), profile location (prof_loc), cloud layer heights at 4 sec (cld_hts), PBL height at 4 sec (pbl_ht), DEM (atm_dem), and time to create elevated aerosol layer heights at 4 and 20 sec (aer_hts).

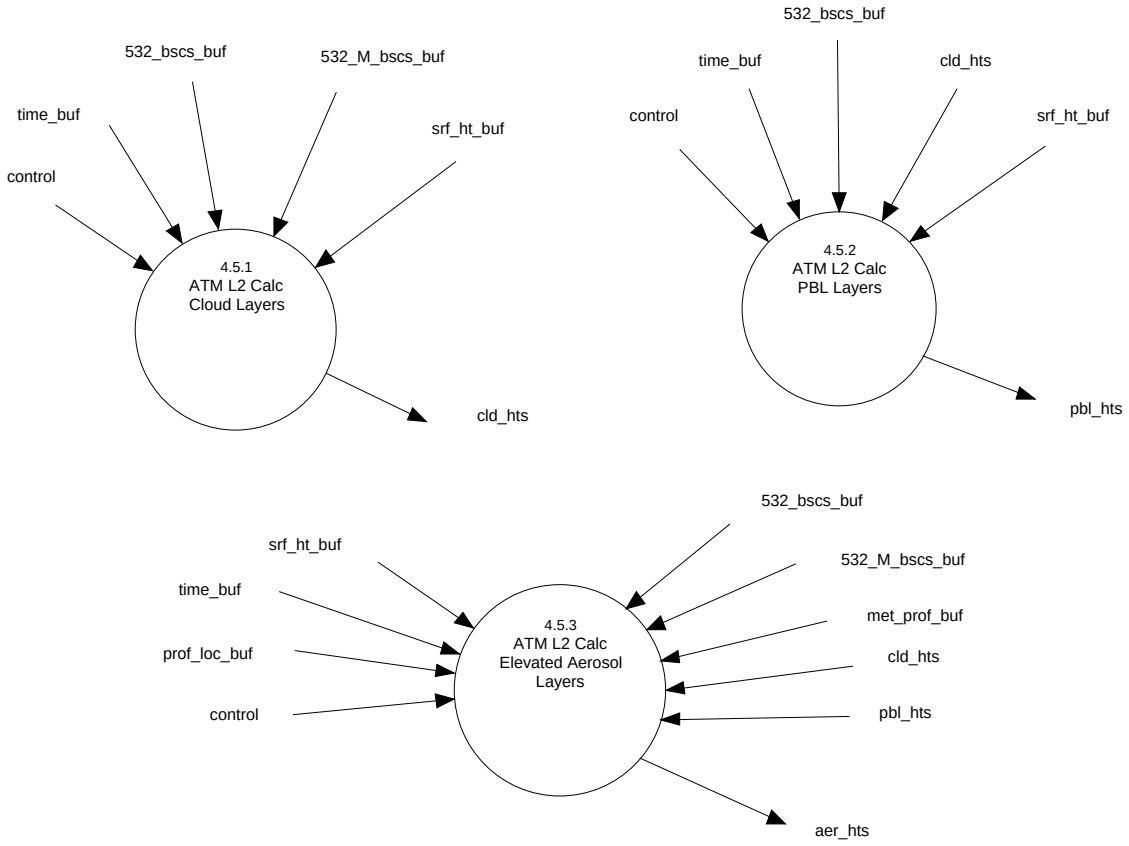

Figure 9-4 Atmosphere Subsystem: Cloud / Aerosol Layer Heights Subprocesses

The ATM L2 Calculate Optical Properties process is divided into two subprocesses. The Calculate Cloud Optical Properties is called from the aerosol routine due to its dependence on aerosol output products. The following is a description of each sub-process.

9.3.14 ATM L2 Calculate Aerosol Optical Properties (A_aer_opt_prop)

Utilizes attenuated backscatter cross section profiles at 532 nm at 5 Hz (532_bscs), met/std atm profiles (met_prof), profile location (prof_loc), molecular backscatter cross section profiles at 532 nm (532_M_bscs), cloud layer heights at 4 sec (cld_hts), PBL height at 4 sec (pbl_ht), aerosol elevated layer heights at 4 and 20 sec (aer_hts), and time to create aerosol backscatter (aer_bs) and extinction cross section profiles at 4 sec (aer_ext), elevated aerosol optical depths at 4 sec (aer_od), and the PBL optical depth at 4 sec (pbl_od).

9.3.15 ATM L2 Calculate Cloud Optical Properties (A_cld_opt_prop)

Utilizes attenuated backscatter cross section profiles at 532 nm at 5 Hz (532_bscs), met/std atm profiles (met_prof), profile location (prof_loc), molecular backscatter cross section profiles at 532 nm (532_M_bscs), cloud layer heights at 1 sec (cld_hts),

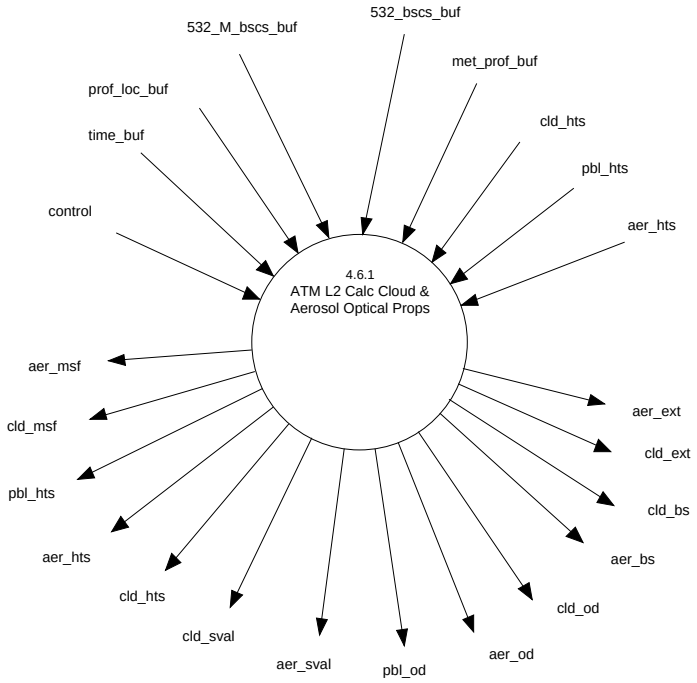

Figure 9-5 Atmosphere Subsystem: Optical Properties Subprocesses

PBL height at 4 sec (pbl_ht), aerosol elevated layer heights at 4 sec (aer_hts), and time to create cloud backscatter (cld_bs) and extinction cross section profiles at 1 sec (cld_ext) and cloud optical depths at 1 sec (cld_od).

Section 10

Level 1B and 2 Elevation Processes

10.1 Function

The Levels 1B and 2 Elevation Computation subsystem generates all elevation Standard Data Products, associated Processing Quality Assessment data, and related computations.

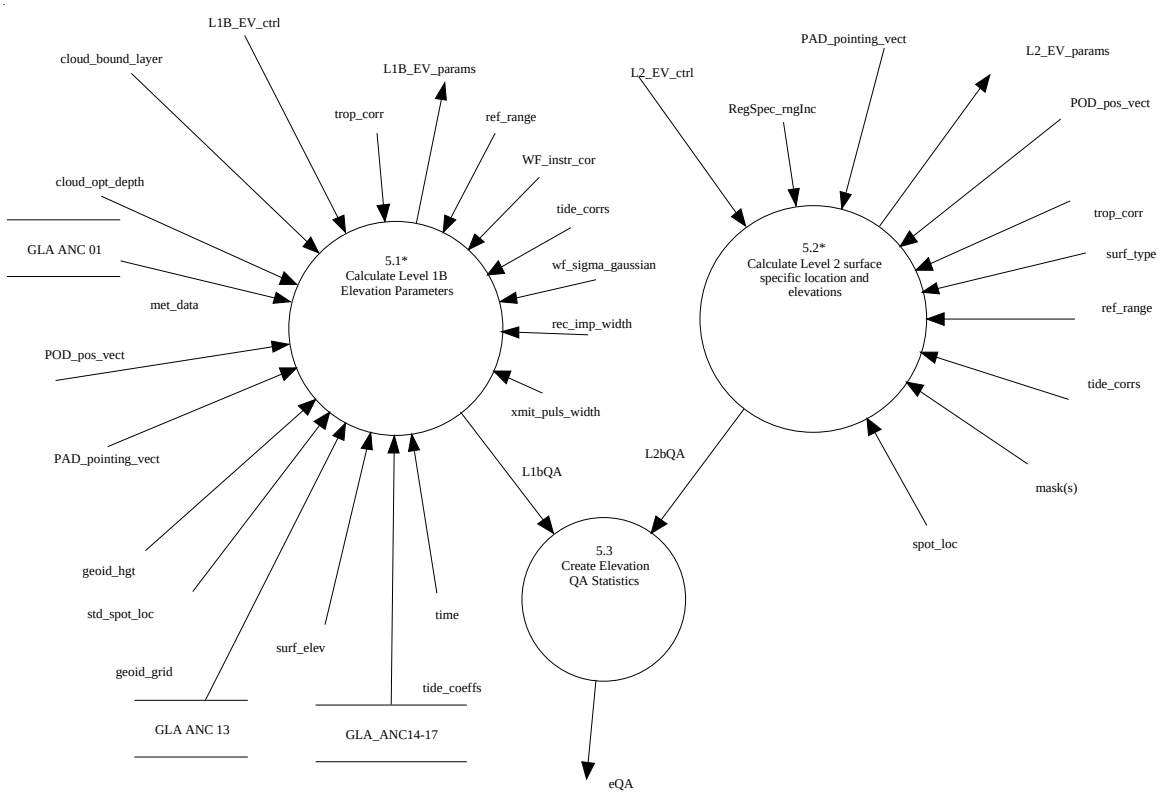

Figure 10-1 Level 1B and 2 Elevation DFD

10.2 Design Constraints / Decisions / Assumptions

- Data will be passed to the Elevation manager one 1-second frame at a time.
- Monitoring, Trend, and QA will be implemented in V2.

10.3 DFDs and their Descriptions

10.3.1 Calculate Level 1B Elevation Parameters

The Level 1B subsystem creates parameters for a Level 1B time-ordered global product (GLA06_SCF) with a geodetically corrected surface-independent standard elevation.

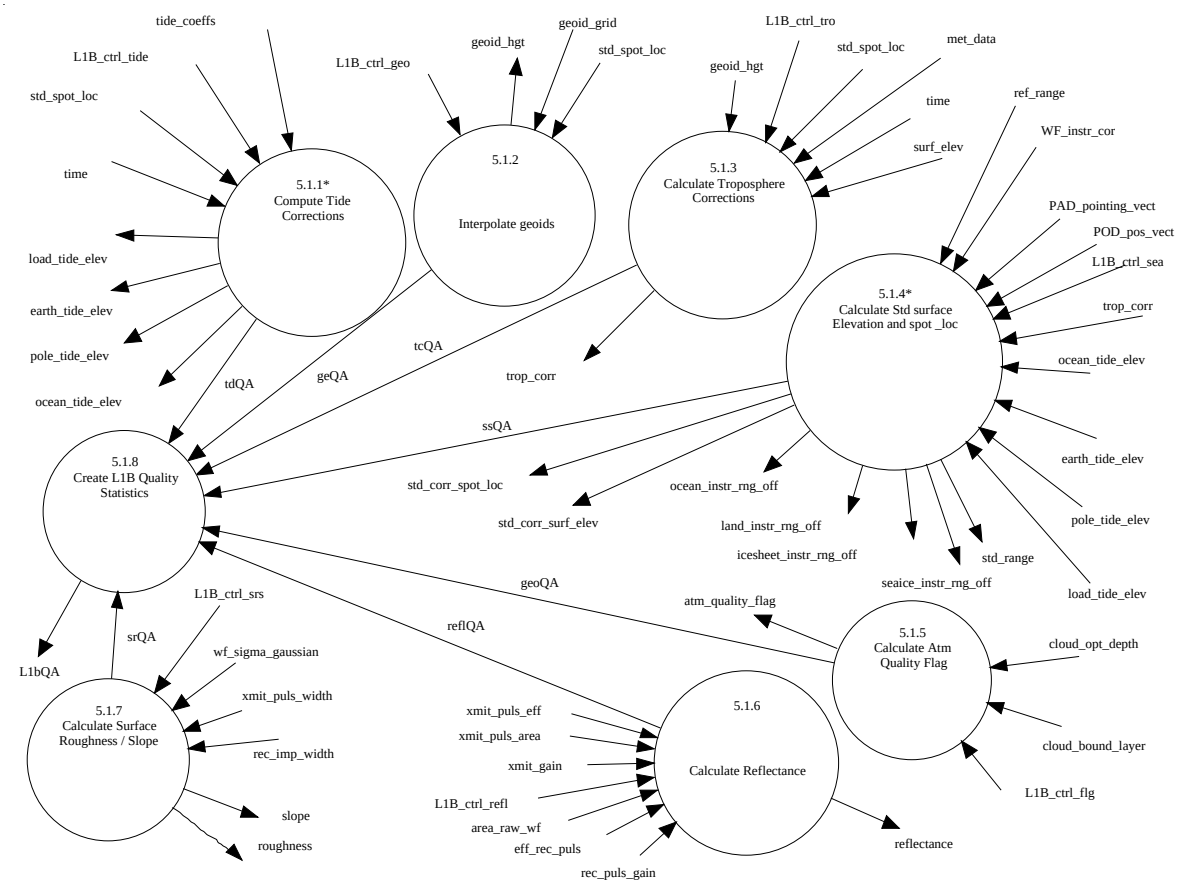

Figure 10-2 Level 1B Elevation Computation DFD

10.3.2 Calculate Level 2 Surface Specific Location and Elevations

The Level 2 subsystem determines region specific (ice sheet, sea ice, land, and ocean regions) elevation parameters for Level 2 time-ordered regional products (GLA12_SCF, GLA13_SCF, GLA14_SCF, and GLAS15_SCF).

Decompositions

Each process is triggered by a control flag which indicates original processing or reprocessing. A description of each of the processes is presented below:

10.3.3 Compute Tide Corrections (E_getTides)

Utilizes the tide coefficients (tide_coeffs), spot location (std_spot_loc) and time (time) to calculate the appropriate tide corrections (load_tide_elev, ocean_tide_elev, and earth_tide_elev).

10.3.4 Interpolate Geoids (E_GetGeoid)

Utilizes the geoid data (geoid_grid) and spot location (std_spot_loc) to interpolate for the Geoid height (geoid_hgt).

10.3.5 Calculate Troposphere Corrections (E_CalcTrop)

Utilizes the met data files (met_data), spot location, and elevation (with respect to the geoid), to interpolate spatially for parameters used in the calculation of the tropospheric corrections (trop_corrs). These corrections are then temporally interpolated to get the tropospheric corrections for the given time.

10.3.6 10.3.6 Calculate Std surface Elevation and spot loc (C_ CalcSploc)

Utilizes the results from the previous three processes along with the spacecraft position in ITRF (POD_pos_vect), the laser attitude in ITRF (PAD_pointing_vect), reference range (ref_range), and waveform parameters (WF_instr_corr) to calculate surface independent elevations (std_corr_surf_elev)and spot locations (std_corr_spot_loc).

10.3.7 Interpolate DEM (E_CalcDEM)

Utilizes the spot locations (latitude and longitude), the Global DEM file, and the DEM mask file to determine the DEM elevation for the specified spot.

10.3.8 Calculate Quality Flag (E_AtmQF)

Utilizes the cloud optical depth (cloud_opt_depth) and cloud boundary layer height (cloud_bound_layer) to ascertain the effect of atmospheric problems that would decrease quality of the elevation product. This quality is returned as a flag (geo_quality_flag).

10.3.9 Calculate Slope & Roughness (E_CalcSlope)

Utilizes the sigma of the gaussian waveform (wf_sigma_gaussian), transmitted pulse width (xmit_puls_width), and receiver impulse width (rec_imp_width) to calculate the slope and roughness.

10.3.10 Calculate Reflectance (E_CalcRefl)

Utilizes the receiver pulse gain (rec_puls_gain), efficiency of the received pulse (eff_rec_puls), area of raw waveform (area_raw_wf), transmit gain (xmit_gain), transmitted pulse efficiency (xmit_puls_eff), and area of transmitted pulse (xmit_puls_area) to calculate the reflectance (reflectance).

10.3.11 Create L1B Quality Statistics

Combines QA data from the previous six processes to create QA statistics for the Level 1B Elevation Computation subsystem

The Compute Tide Corrections process consists of three subprocesses (5.1.1.1 - 5.1.1.3) which calculate the elevation corrections due to the effects of the load tide, ocean tide, and earth tide. Each subprocess is triggered by a control flag.

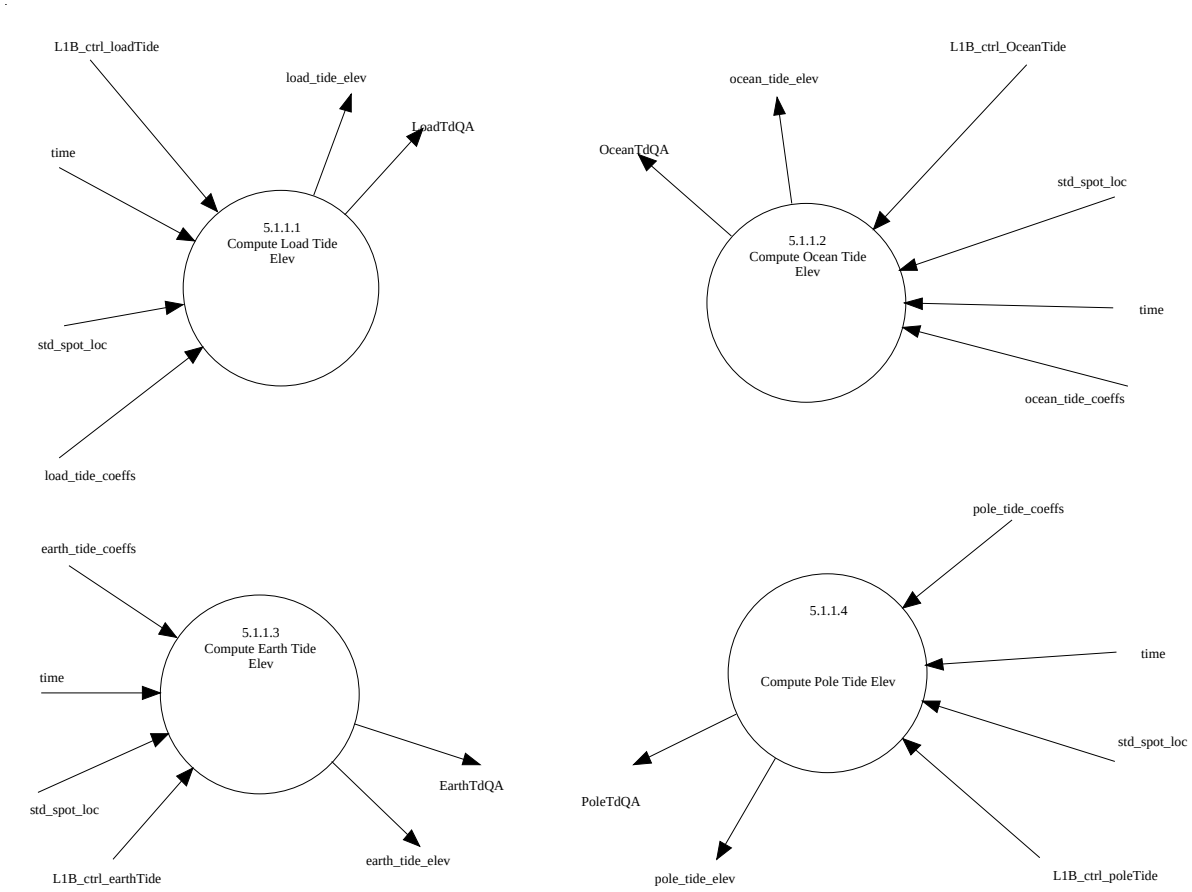

Figure 10-3 Compute Tide Corrections DFD

The following is a brief description of each of the subprocesses.

10.3.12 Compute Load Tide Correction (E_calcLoadTd)

Utilizes the load tide coefficients file (load_tide_coeffs) to compute the coefficients for the given spot location (std_spot_loc). Then calculates the load tide correction (load_tide_corr) using the given time (time).

10.3.13 Compute Ocean Tide Correction (E_calcOceanTd)

Utilizes the ocean tide coefficients file (ocean_tide_coeffs) to compute the coefficients for the given spot location (std_spot_loc). Then calculates the ocean tide correction (ocean_tide_corr) using the given time (time).

10.3.14 Compute Earth Tide Correction (E_calcEarthTd)

Utilizes the earth tide coefficients file (earth_tide_coeffs)to compute the coefficients for the given spot location (std_spot_loc). Then calculates the earth tide correction (earth_tide_corr) using the given time (time).

The Level 2 Elevation Computation subsystem consists of five processes (5.2.1 - 5.2.5).

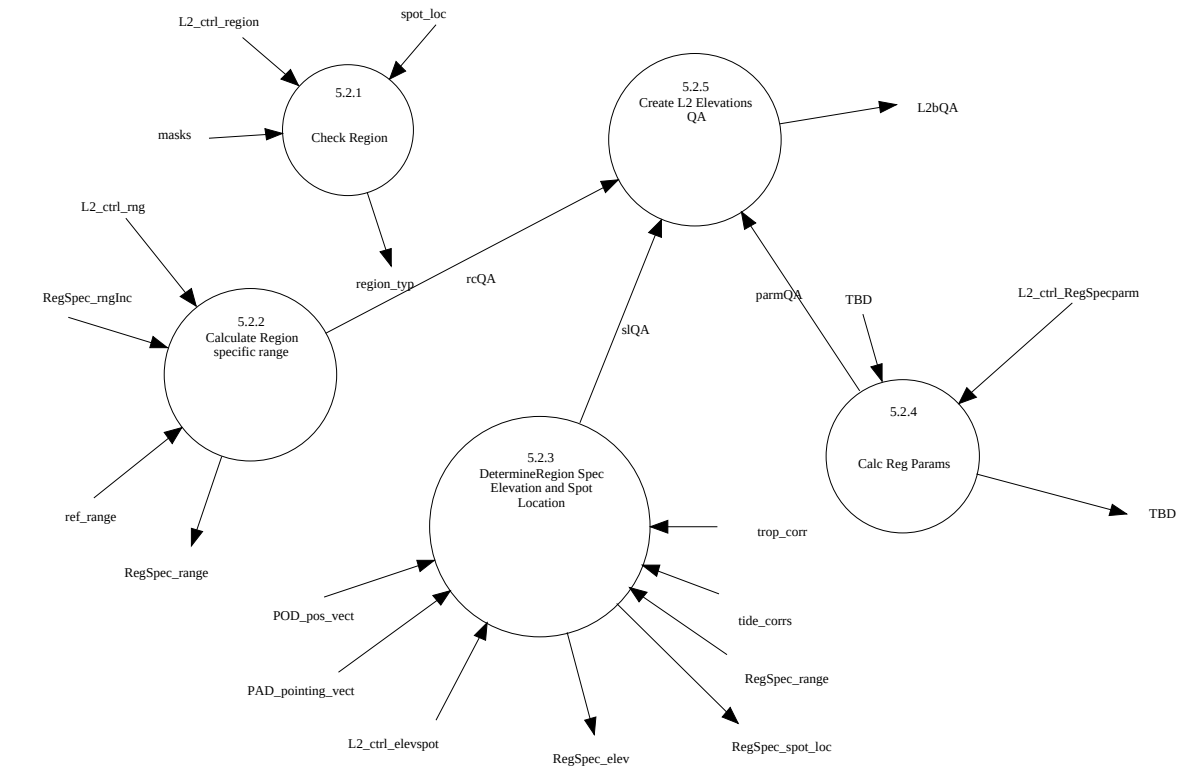

Figure 10-4 Calculate Level 2 Elevations DFD

A description of each of the processes is presented below.

10.3.15 Check Region (E_ChkReg)

Utilizes the region masks (masks) and spot location (spot_loc) to determine if the data lies in a valid region (region_typ).

10.3.16 Determine Region Spec Elevation and Spot Location (C_CalcSploc)

Utilizes the region specific range (RegSpec_range), POD position vector (POD_pos_vect), PAD vectors (PAD_pointing_vect), tide corrections (tide_corrs), and trop corrections (trop_corr) to calculate the region specific elevation (RegSpec_elev) and spot location (RegSpec_spot_loc).

10.3.17 Calc Reg Params (E_SeaIceParm, E_OceanParm,E_LandParm)

Calculates the region specific parameters that have not already been determined earlier by the other routines. Inputs and outputs are TBD.

10.3.18 Create L2 Elevations QA (E_CreateL2QA)

Combines QA data from the processes 5.2.2, 5.2.3, and 5.2.4 (C_CalcRange, E_CalcSploc, E_CalcRegParm) to create QA statistics for the Level 2 Elevation Computation subsystem.

Section 11

State Transition Diagrams

The Dynamic Model provides another view of GSAS and highlights typical scenarios that the system will execute. GLAS_Exec may be regarded as a State Machine that assumes a series of states depending on inputs to the system and the actions that occur in a particular state. Each execution is a scenario. Since all possible scenarios will not be diagrammed, decision tables provide the remaining possible execution paths and also the required actions when the system is in reprocessing mode.

11.1 GLAS_Exec

The GLAS_Exec process moves from its initial state to one of the possible new states characterized by processing in a particular subsystem. The four states are:

- Level 1A Computations,
- Level 1B Waveforms,
- Level 1B and 2 Atmosphere and
- Level 1B and 2 Elevation.

I-SIPSCtrl represents the trigger that determines the execution path for the software. The software proceeds to the end state, which completes the execution, or the any of the possible states that can be accessed in the direction of the arrows shown in Figure 11-1.

The GLAS_Exec decision table (Table 11-1) indicates that four major subsystems can be executed and, for any change in one of the subsystems, which processes will be affected and will required to be executed.

Table 11-1 GLAS_Exec Decision Table

Variables	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Change in L1A Computations	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1
Change in L1B Waveforms	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
Change in L1B and 2 Atmosphere	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
Change in L1B and 2 Elevation	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1
Action															
Execute L1A Processes	x	x		x		x		x		x		x		x	x

Table 11-1 GLAS_Exec Decision Table (Continued)

Execute L1B Waveform processes	x		x	x			x	x			x	x			x
Execute L1B and 2 Atmosphere Processes	x				x	x	x	x					x	x	x
Execute L1B and 2 Elevation Processes	x								x		x	x	x	x	x

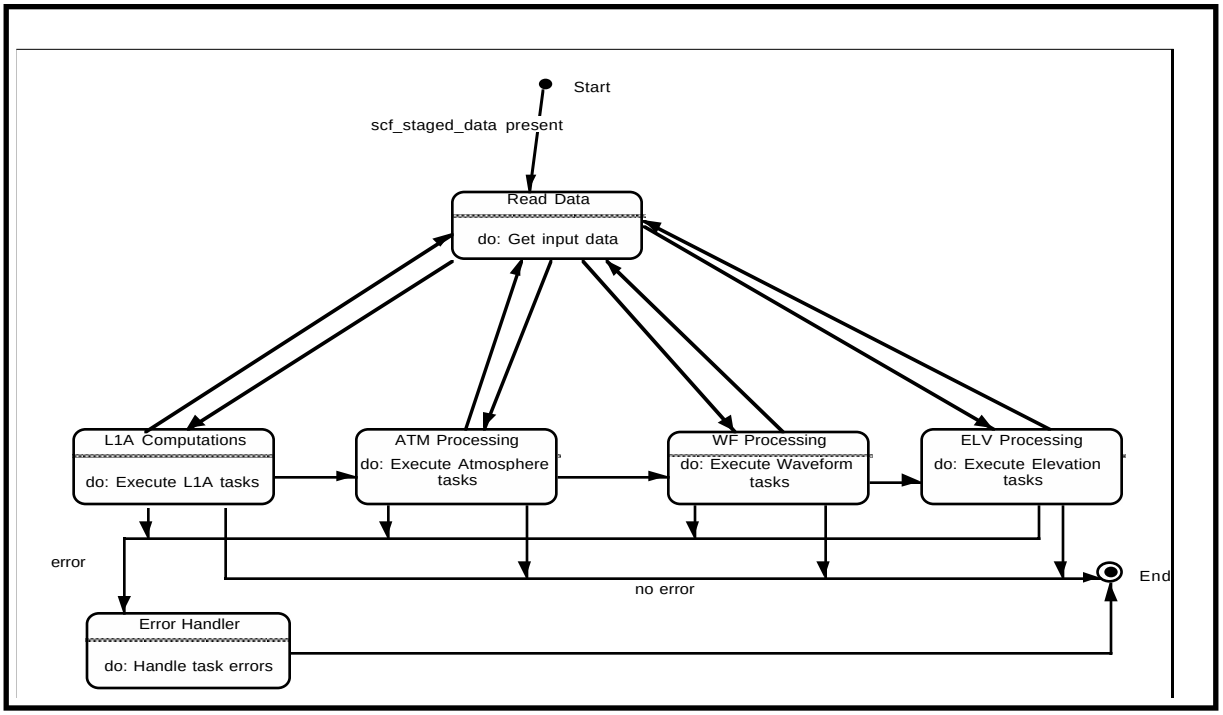

Figure 11-1 Execute a Task

The state diagram (Figure 11-1) for executing a specific task shows all possible scenarios and highlights the Read Data state, which is responsible for reading the input data one record at a time. For each record, GLAS_Exec transitions through the 4 subsystem states and the error handler, if there is an error in any subsystem. If the error is fatal, the process ends. The system returns to the read state after each record is processed. When all records are read and processed, the system proceeds to the end state.

11.2 Level 1A Computations

The Level 1A Computations process transitions through the 8 states shown in the state transition diagram. From the initial state, it moves to the calc_shot_time state where it computes the 40 per second shot time then it moves to the L_GetPredLoc state where the predicted location is determined. The process then moves to the L_Eng_proc state and processes engineering data. In the L_Alt state the altimeter

data is processed. In the L_LID_proc state the L1A Lidar data is processed. In the L_Loc_Att_proc state, instrument data, spacecraft location and attitude data are collected. In the L_QA_Trnd state L1A quality assurance and trending analysis is performed before the process terminates in the end state.

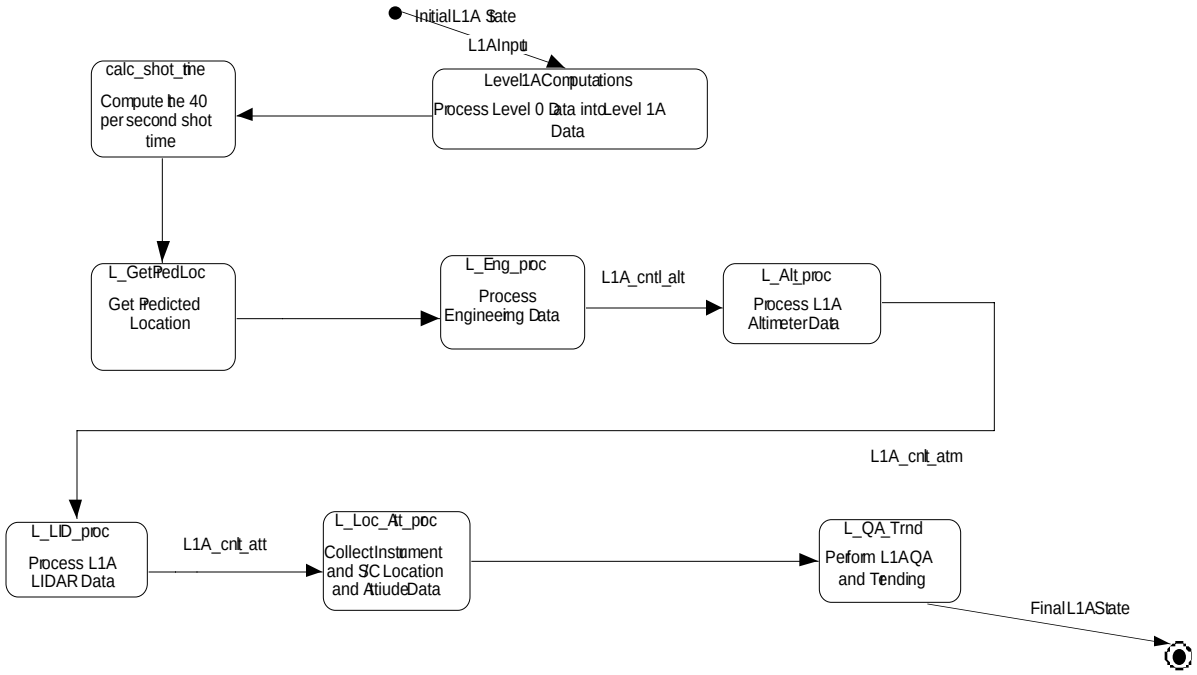

Figure 11-2 L1A Computations

The transitions occur for each record and may occur sequentially through all 4 states. Each transition depends on the control received by the subsystem for GLAS_Exec. When all records have been processed, state transition occur from the Level 1A Computations to the L_QA_Trnd where trend analysis data are created. The Final state completes the execution.

The state diagram shows that the L_Eng is called for every scenario. Then, depending on the control values, the other ATBDs are called. Each of the ATBDs returns to the manager. The QA/Trend processing is called last.

The variables in the Level 1A processing are updated inputs or updated software. Updated inputs will occur if missing packets are recovered or ancillary data is updated. Updated software will occur if algorithms are modified or code errors are corrected.

An entire ATBD will be executed for any update related to that ATBD. The QA Stats process will always be executed if an ATBD is re-run.

The 4 individual scenarios are:

1) The input to the altimeter ATBD is updated or the ALT ATBD software is modified.
2) The input to the atmosphere ATBD is updated or the ATM ATBD software is modified.
3) The input to the engineering ATBD is updated or the ENG ATBD software is modified.
4) The input to the attitude ATBD is updated or the ATT ATBD software is modified.

The decision table lists all the possible combinations of individual scenarios. Since some of the processing in the ALT, ATM, and ATT ATBDs is dependent on engineering data that is output from the ENG ATBD, the ENG ATBD will be executed first and any time the engineering data is re-processed the ALT, ATT, and ATM ATBDs will be executed. The Level 1A data will always be re-created from the Level 0 data. Therefore, for any reprocessing scenario the Level 1A engineering ATBD will always be executed since the other ATBDs rely on Level 1A engineering data as input. The ALT, ATM, and ATT ATBDs are not dependent on each other and can be executed in any order or concurrently.

Table 11-2 Decision Table for the Level 1A Computations

Var	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ALT ATBD input or code updates	Y	Y	Y	Y	N	N	N	N	N	Y	N	Y	N	Y	Y	N
ATM ATBD input or code updates	N	Y	Y	Y	Y	Y	Y	N	N	N	N	N	Y	N	Y	N
ENG ATBD input or code updates	N	N	Y	Y	N	Y	Y	Y	Y	Y	N	N	N	Y	N	N
ATT ATBD input or code updates	N	N	N	Y	N	N	Y	N	Y	Y	Y	Y	Y	N	Y	N
Actions																
Do ALT ATBD	X	X	X	X		X	X	X	X	X		X		X	X	
Do ATM ATBD		X	X	X	X	X	X	X	X	X			X	X	X	
Do ENG ATBD	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
Do ATT ATBD			X	X		X	X	X	X	X	X	X	X	X	X	
Do QA Stats	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

11.3 Level 1B Waveforms

The normal sequence of states is indicated with double arrows. The waveforms will be assessed and the standard range corrector will be calculated (W_Assess); and then other waveforms will be parameterized (W_FunctionalFt). This sequence will continue for each record. Quality Assurance (QA) statistics will be created and returned when the waveform manager (WF_Manager) calls W_CreQAStats. All waveform subprocesses will call GLAS_Error if an error is detected, but it will not be shown on any other diagram.

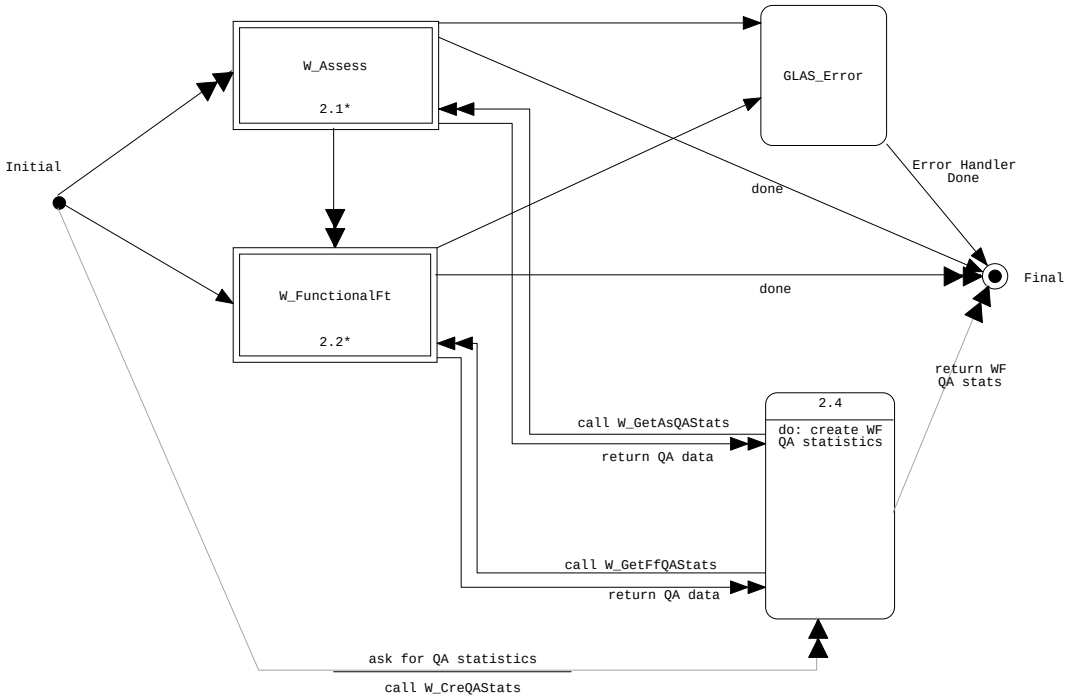

Figure 11-3 Level 1B Waveforms State Diagram

11.3.1 Assess Waveforms & Calculate Standard Range Corrector

The normal sequence of states (indicated by double headed arrows) is: calculate the relative time (W_calcRelTime); characterize the transmitted pulse (W_CharTrPulse); calculate the background noise and standard deviation of the background noise (W_CalcNoise); calculate the reference range (W_CalcRefRng); calculate the smoothed waveform and determine the signal begin and end, and the preliminary uncorrected range offset (W_SmoothPreRC); determine the geolocation (W_DetGeo); get the regions (C_GetRegions); check the waveform for saturation (W_Ck4Sat); calculate the centroid, maximum, area, and asymmetry of the waveform (W_CalcCtMxArAs); and then calculate the threshold retracker range offset (W_CalcThRetrkr). Quality assurance (QA) information will be stored in

W_Assess_mod and returned to the waveform manager when it calls W_GetAsQAStats.

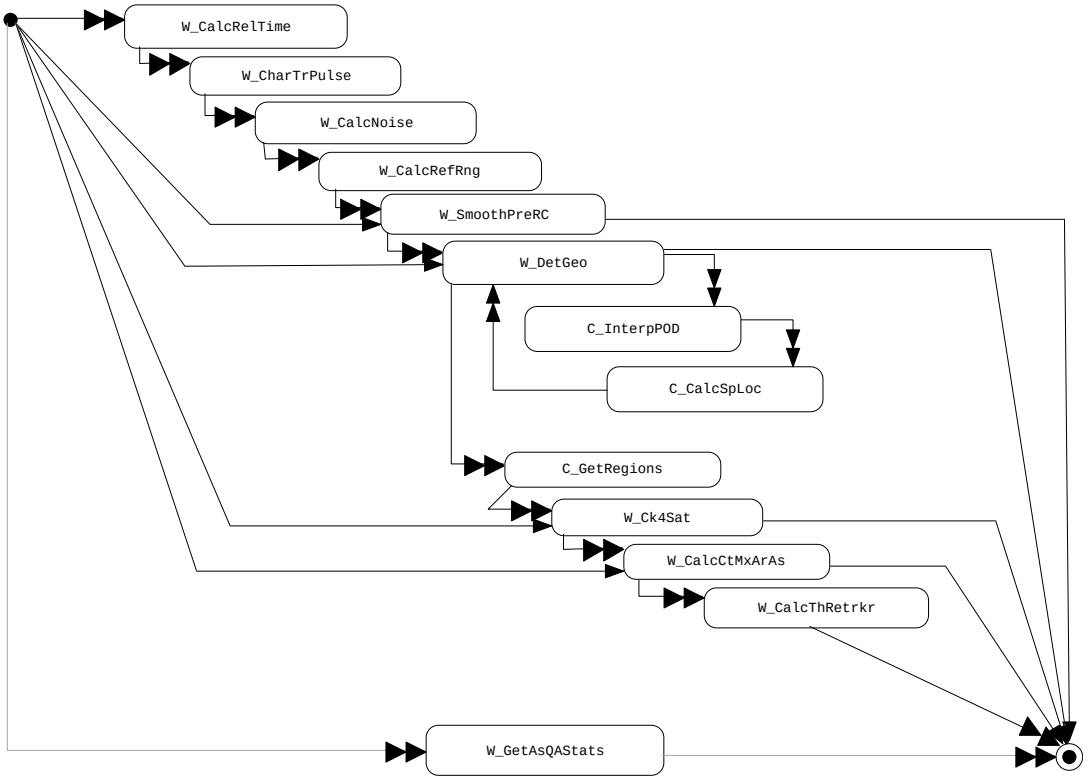

Figure 11-4 Access Waveforms & Calculate Standard Range Offset State Diagram

11.3.2 Calculate Other Waveform Characteristics

The normal sequence of states (indicated by double headed arrows) is: parameterize the waveform (W_ParamWithFit). Quality assurance (QA) information is stored in

W_FunctionalFt_mod and returned the waveform manager when it calls W_GetOtQAStats.

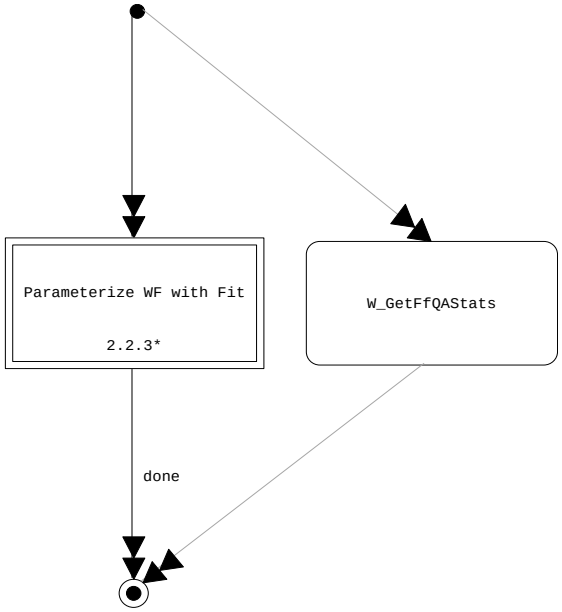

Figure 11-5 W_Functional Fit State Diagram

Table 11-3 Reprocessing Decision Table

If these variables change:	1	2	3	4	5
land algorithm	x				
non-land algorithm		x			
orbit (POD)			x		
attitude (PAD)				x	
region type grid					x
Then these actions are taken:					
1.1.4.1 Assess waveforms (W_Assess)	x	x	x	x	x
1.1.4.1.6 Calculate Elevation & Geolocation (W_DetGeo)			x	x	

Table 11-3 Reprocessing Decision Table (Continued)

1.1.4.1.11 Determine Region Type (C_GetRegions)			x	x	x
1.1.4.2 Perform Functional Fit (W_FunctionalFt)	x	x	x	x	x

11.4 Level 1B and 2 Atmosphere Computations

The accompanying state transition diagram illustrates the atmosphere computations processing states. The first diagram shows the initial state (start of processing) leading into the atmosphere subsystem manager (AtmMgr). From the manager, arrows indicate each of the reprocessing scenarios. The normal end-to-end processing sequence (1) starts with process C_interp_pod. Processes that do not have arrows leading directly to them from the manager cannot be the first processes called in a reprocessing scenario. This occurs in instances when processes need input data that are not saved, therefore that data must be regenerated in a prior process for that process to use. For instance, it is shown that scenarios 2, 4, and 5 start with process A_interp_met. This is because the interpolated met data, which are used in several processes, are not saved for reprocessing access and therefore must be regenerated at the start of these reprocessing scenarios.

Double headed arrows (marked with the number 1) indicate the main processing sequence, while numbered lines with single arrows represent partial reprocessing scenarios. All of the processing scenarios eventually end at the final state (end of processing). In normal production processing, processes C_interp_pod through WriteATM are triggered in the order shown on the diagrams from left to right. In partial reprocessing, the processes are still executed in that order, however, the initial process may be different and some intermediate processes may be skipped. The line numbers indicate the scenarios that they represent. These scenarios correspond with the sequence of actions outlined in the state transition table above. For example, when scenario 5 is executed, the first process called is A_interp_met, then A_buff_data, A_cld_lays, A_pbl_lay, A_aer_lays, A_aer_opt_prop, A_qa_G8-11, and WriteATM are executed. This processing scenario corresponds with the sequence executed by the “cld_to_end” control outlined in the state transition table below.

The accompanying state transition table illustrates the possible scenarios for atmosphere data processing.

“Controls” are from an input control file and indicate which processing scenario is run. There are four controls available for the atmosphere subsystem. The control “backscatter to end” (bs_to_end), represents the complete processing sequence which produces products GLA07 - 11. The control “backscatter only” (bs_only), represents only the processes required to produce the product GLA07. Likewise, the control “cloud to end” (cld_to_end), represents the partial processing sequence which produces only products GLA08 - 11. The control “no_POD” indicates whether POD processing should be run to calculate latitude and longitude for output to GLA07. This is meant for use in the partial reprocessing of GLA07, when algorithms change but the POD does not change, thereby requiring the reprocessing of data, but not the repro-

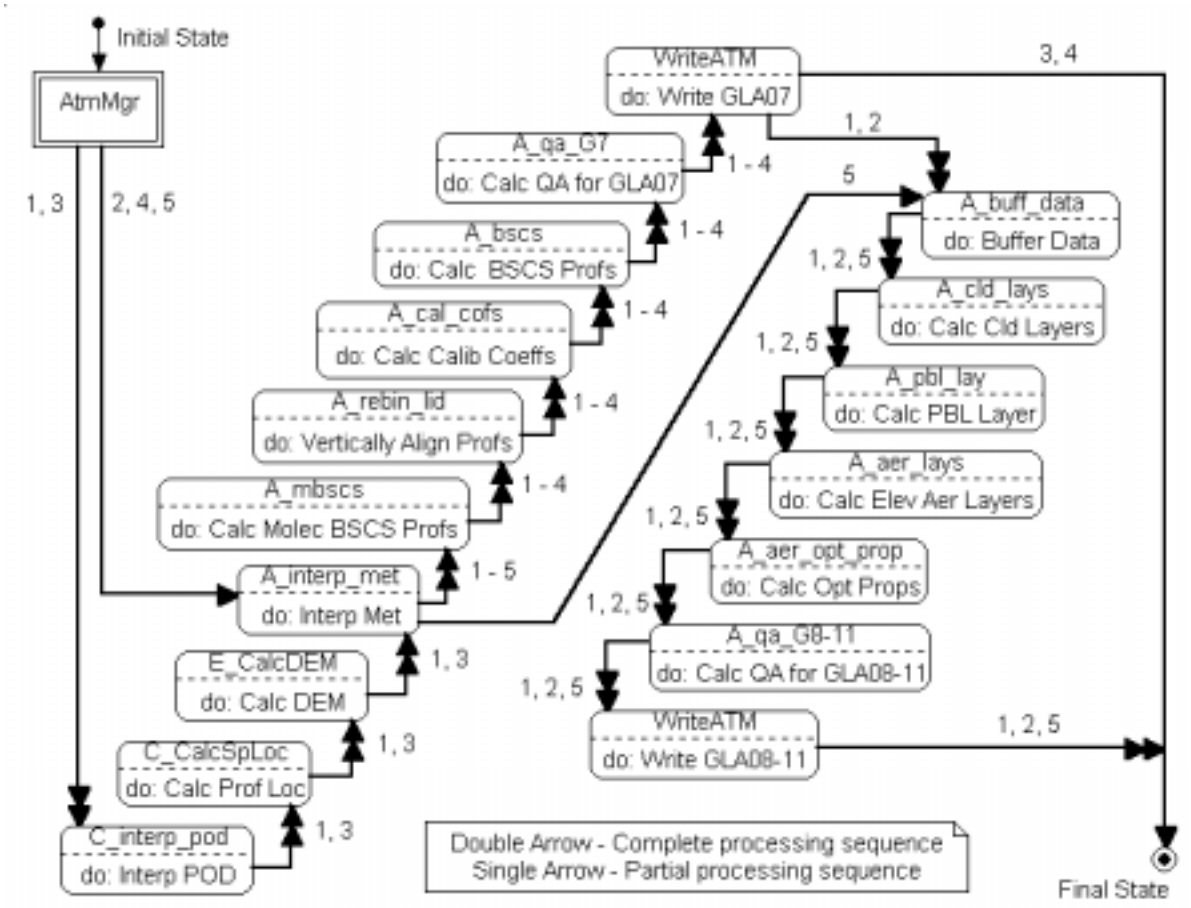

Figure 11-6 Atmosphere Computations

cessing of geolocation. If this control is set to “true”, then the geolocation values already on GLA07 are used.

“Action” indicates whether processes within the atmosphere subsystem are run for a given scenario. “Yes” indicates that the process is run, while “No” indicates that it is not. For instance, if the control “cld_to_end” is set, then processes A_interp_met, A_buff_data, A_cld_lays, A_pbl_lay, A_aer_lays, A_aer_opt_prop, A_qa_G8-11, and WriteATM are executed.

11.5 Level 1B and 2 Elevation

Figures 9 and 10 show the various processing / reprocessing scenarios possible for Level 1B and Level 2 Elevations. The End-to-End state signifies a normal processing scenario. In Figure 9, the normal end-to-end state is as follows: first calculate the Tide corrections i.e., Ocean Tide (E_CalcOceanTd), Earth Tide (E_CalcEarthTd), and Load Tide (E_CalcLoadTd). Then the geoid height (E_GetGeoid), the troposphere correc-

Table 11-4 Atmosphere Computations Decision Table

	Controls					
	no_POD	No	**Yes**	No	**Yes**	No
	bs_to_end	**Yes**	**Yes**	No	No	No
	bs_only	No	No	**Yes**	**Yes**	No
	cld_to_end	No	No	No	No	**Yes**
	Action					
	C_interp_pod	**Yes**	No	**Yes**	No	No
	C_CalcSpLoc	**Yes**	No	**Yes**	No	No
	C_CalcDEM	**Yes**	No	**Yes**	No	No
	A_interp_met	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
	A_mbscs	**Yes**	**Yes**	**Yes**	**Yes**	No
	A_rebin_lid	**Yes**	**Yes**	**Yes**	**Yes**	No
	A_cal_cofs	**Yes**	**Yes**	**Yes**	**Yes**	No
	A_bscs	**Yes**	**Yes**	**Yes**	**Yes**	No
	A_qa_G7	**Yes**	**Yes**	**Yes**	**Yes**	No
	WriteATM (GLA07)	**Yes**	**Yes**	**Yes**	**Yes**	No
	A_buff_data	**Yes**	**Yes**	No	No	**Yes**
	A_cld_lays	**Yes**	**Yes**	No	No	**Yes**
	A_pbl_lay	**Yes**	**Yes**	No	No	**Yes**
	A_aer_lays	**Yes**	**Yes**	No	No	**Yes**
	A_aer_opt_prop	**Yes**	**Yes**	No	No	**Yes**
	A_qa_G8-11	**Yes**	**Yes**	No	No	**Yes**
	WriteATM (GLA08-11)	**Yes**	**Yes**	No	No	**Yes**
	Processing scenario	1	2	3	4	5

tions (E_CalcTrop), geolocation (C_CalcSploc), slope and roughness (E_CalcSlope), and finally the reflectance (E_CalcRefl) values are calculated.

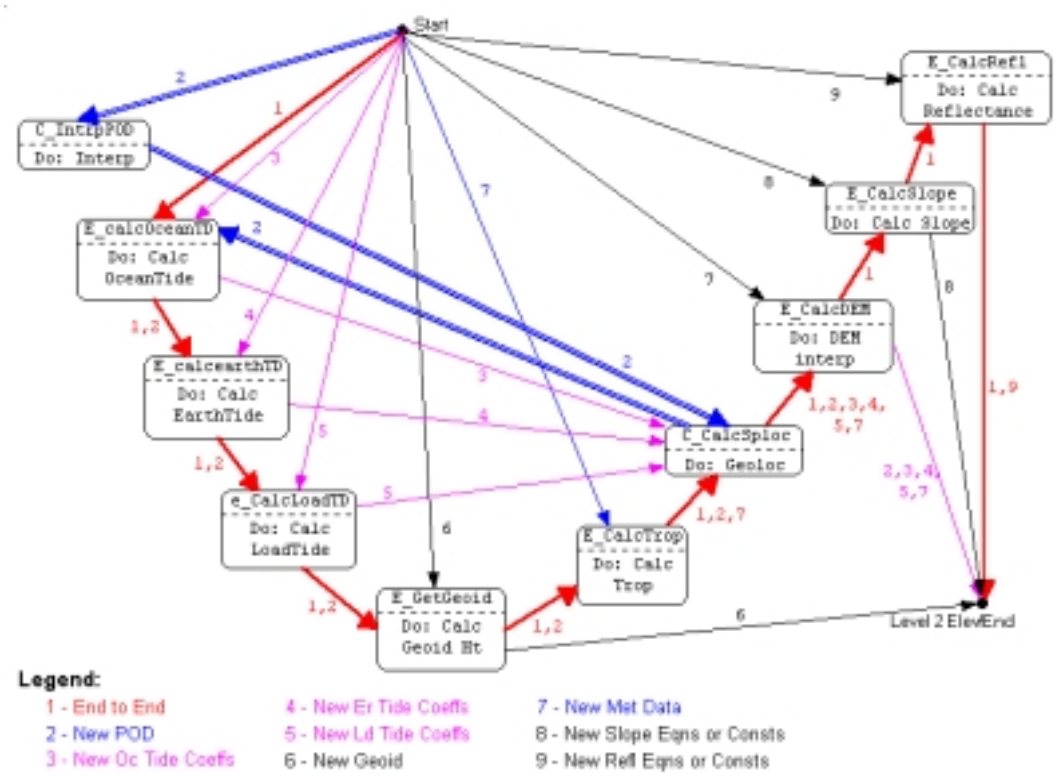

Figure 11-7 Level 1B and 2 Elevation

In the case of the Level 2 elevation (Figure 11-8), an end-to-end process flow comprises of: check to see which region the spot lies in (E_ ChckReg), and then depending on the regions defined and requested calculate the spot locations and elevations (C_CalcSploc). The region specific parameters will then be calculated (E_CalcRegParm).

Table 11-5 shows the reprocessing scenarios that will be considered. Changes in the listed variables (designated as Yes - row wise), will require the processes, listed as Yes, to be executed. The order of processing is sequential (column-wise) as shown.

The scenarios reflect changes in Level 1B and 2 elevations. If the results are only needed for GLA06, we stop at the end of the actions listed under Level 1B elevation. If the results are required for GLA12, 13, 14, 15 we continue till the end of Level 2 elevation.

NOTE: If there are a combination of changes the collective action will be the sum of each individual action. The order of processing is in the order shown in Table 11-5.

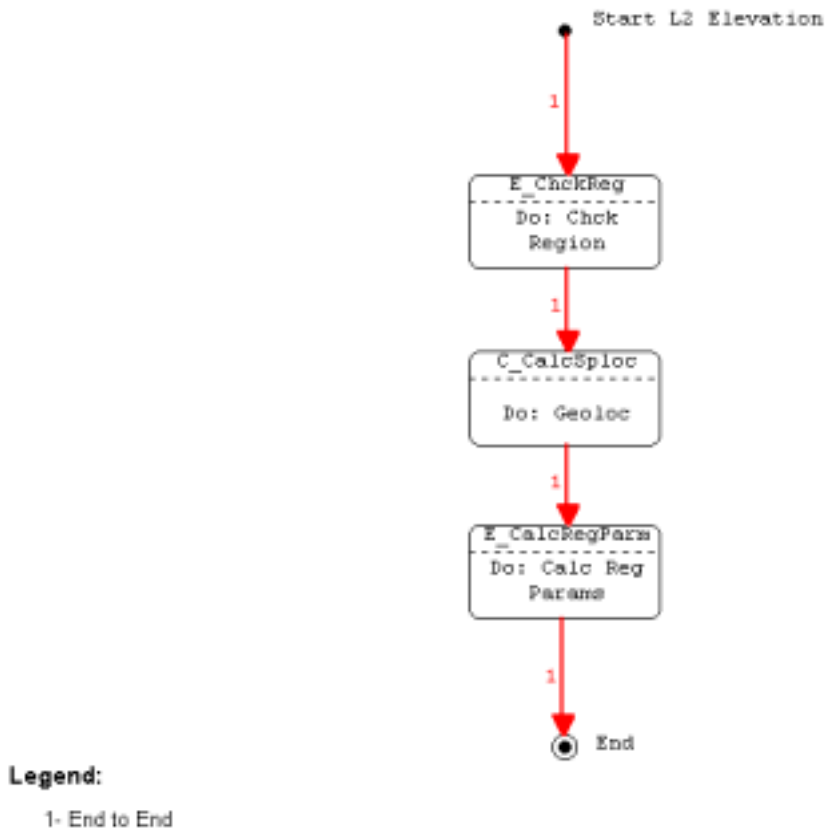

Figure 11-8 Level 2 Elevation – Check Region

Table 11-5 Reprocessing Scenario Stages For Level 1B & 2 Elevations

Variables				State						
POD	Yes									
PAD		Yes								
Ocean Tide Model			Yes							
Earth Tide Model				Yes						
Load Tide Model					Yes					
Geoid Model						Yes				
Met Data							Yes			
DEM Model								Yes		
Slope/Rough Constants									Yes	
Reflection Constants										Yes
Action										

Table 11-5 Reprocessing Scenario Stages For Level 1B & 2 Elevations (Continued)

Variables				State						
Level 1B Elevation										
Interp POD	Yes									
Ocean Tide Corr	Yes		Yes							
Earth Tide Corr	Yes			yes						
Load Tide Corr	Yes				Yes					
Geoid Hgt	Yes					Yes				
Trop Corr	Yes						Yes			
Calc Std Elev & Spot	Yes	Yes	Yes	Yes	Yes		Yes			
Calc DEM	Yes	Yes						Yes		
Calc Slope & Roughness									Yes	
Calc Reflectance										Yes
Create L1B Quality Stats	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes
Level 2 Elevation										
Check region	Yes	Yes	Yes	Yes	Yes		Yes		Yes	Yes
Calc SpotLoc	Yes	Yes	Yes	Yes	Yes		Yes		Yes	Yes
Calc Reg Parms	Yes	Yes	Yes	Yes	Yes		Yes		Yes	Yes

Section 12

Structure Model (Structure Charts)

The structure model provides a hierarchical view of the GSAS system and identifies the modules that will be implemented. The model is derived from the Process Model through analysis and translation of the processes into modules.

GLAS_Exec is the main program which call all the input modules. It initializes data structures and executes the ATBD algorithms sequentially, on record at a time. The submanager executes the science algorithms and writes the products.

Common modules such at the error handler and the spot location calculations are developed as utilities for other modules to use.

The major arrows leaving a module indicate a program call to the module to execute a task or function. As far as possible, variable arguments have been identified for each module and are represented as input or output variables, with arrows along the major arrows pointing in or out of the modules respectively.

12.1 GLAS_Exec

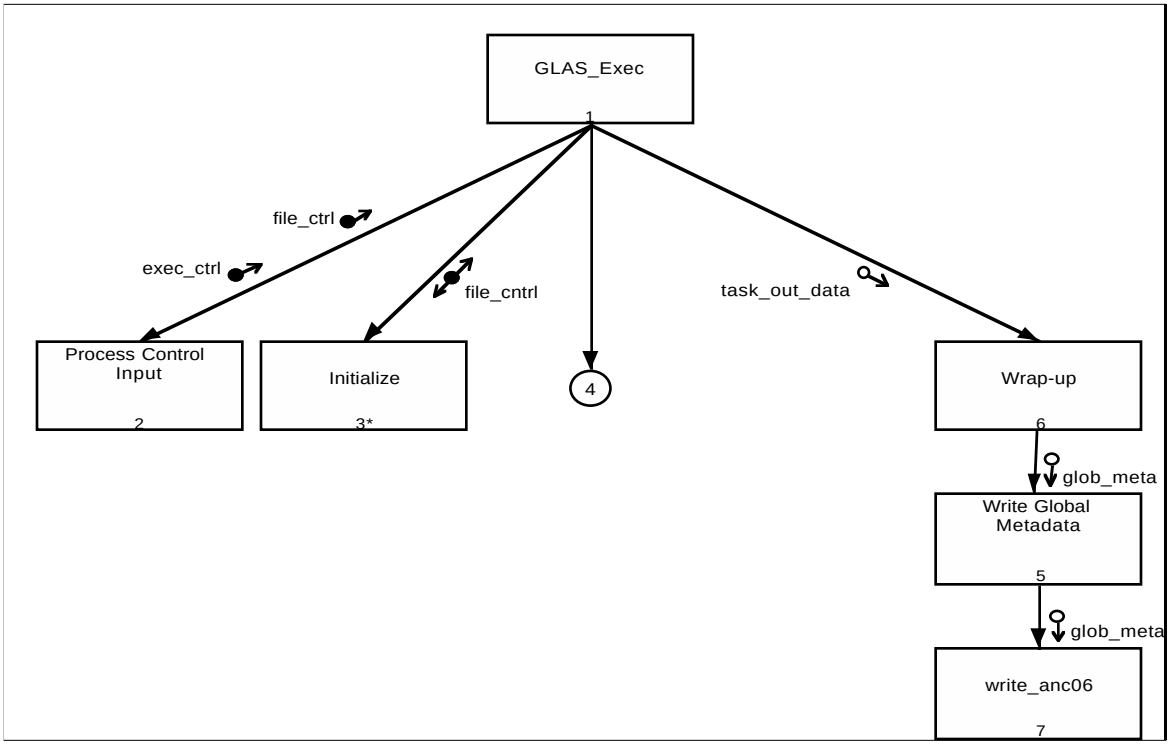

Figure 12-1 GLAS Exec

GLAS_Exec calls Process Control Input, Initialize, the science ATBD implementations depicted by the stub labeled 4, and wrap-up. Each of these modules do make other

calls to lower level modules. Wrap-up calls Write Global Metadata, which in turn calls write_anc06.

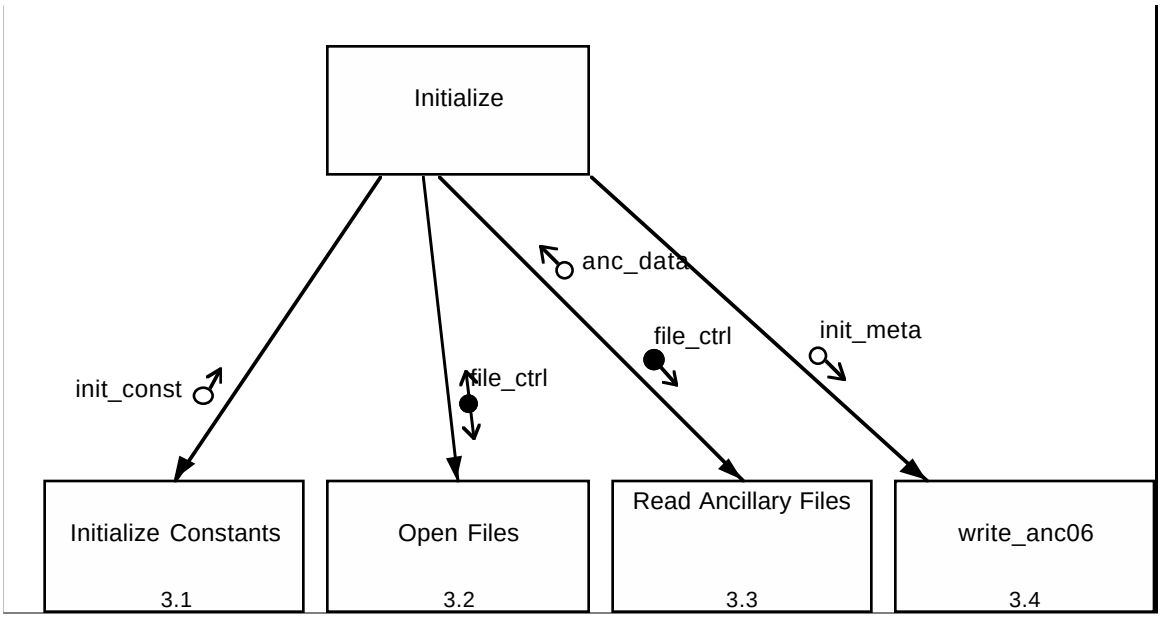

Figure 12-2 Initialize

Initialize calls 4 main modules as shown above.

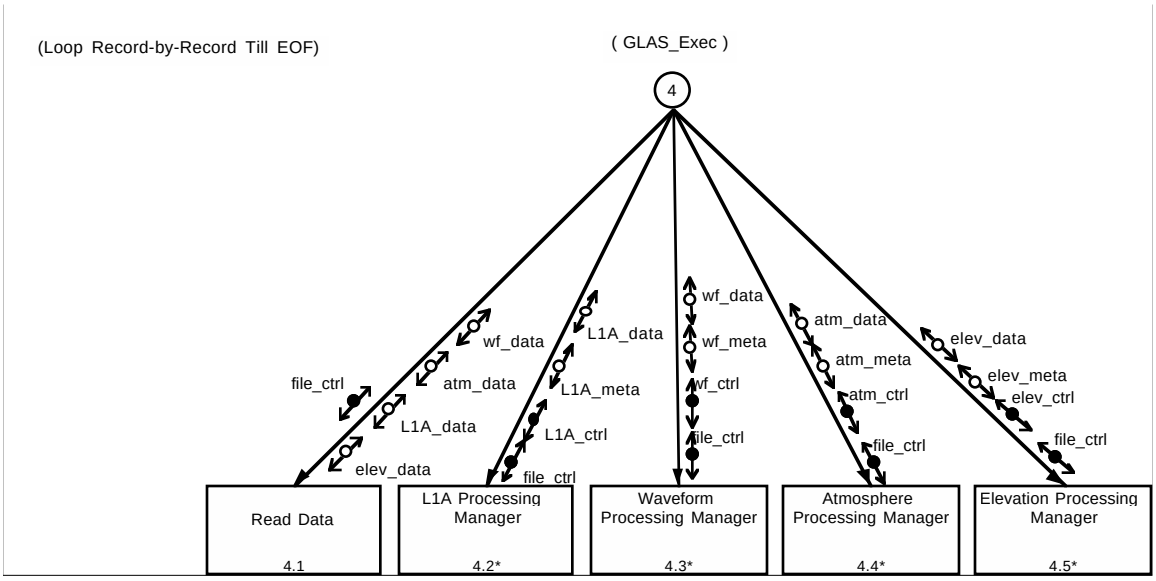

Figure 12-3 Managers Stub (4)

GLAS_Exec calls Read Data for each record which it processes with sequential calls to the L1A Processing Manager, Waveform Processing Manager, Atmosphere Pro-

cessing Manager and Elevation Processing Manager. Calls to the modules below the submanagers are shown in the succeeding diagrams.

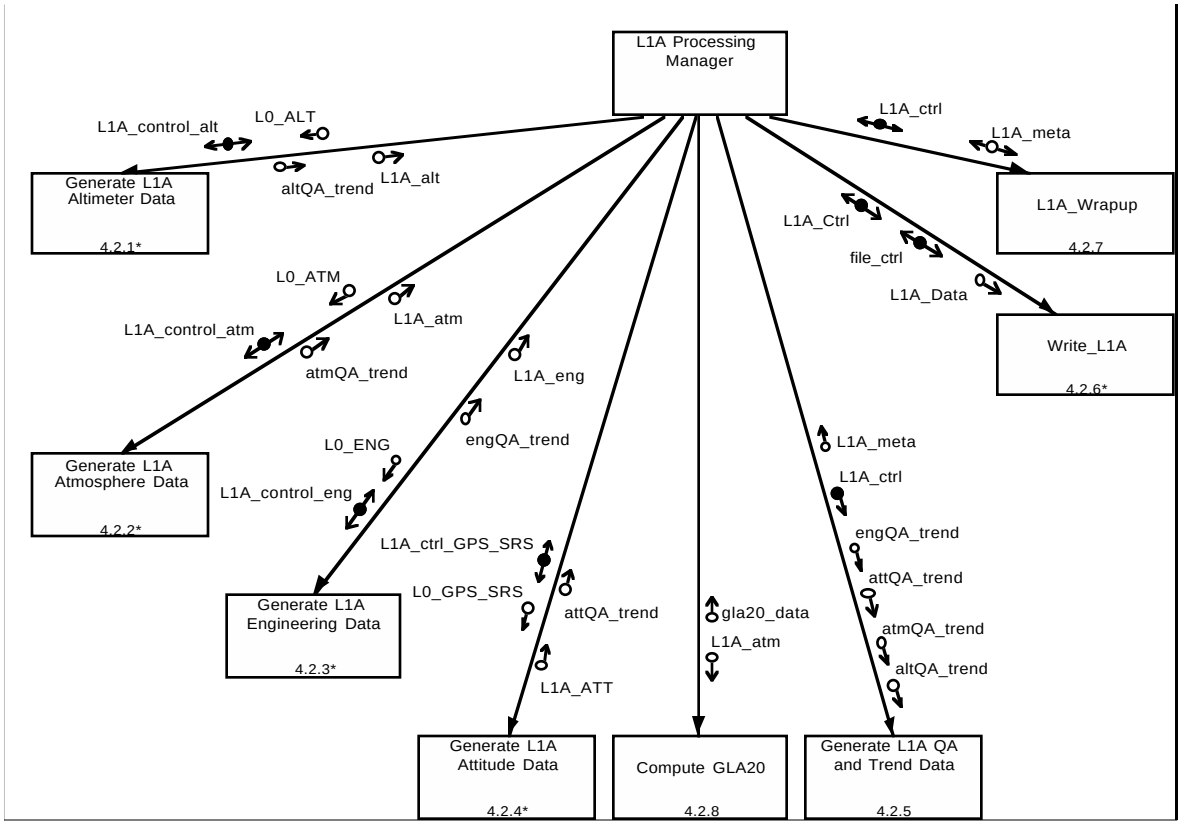

Figure 12-4 L1A Processing Manager

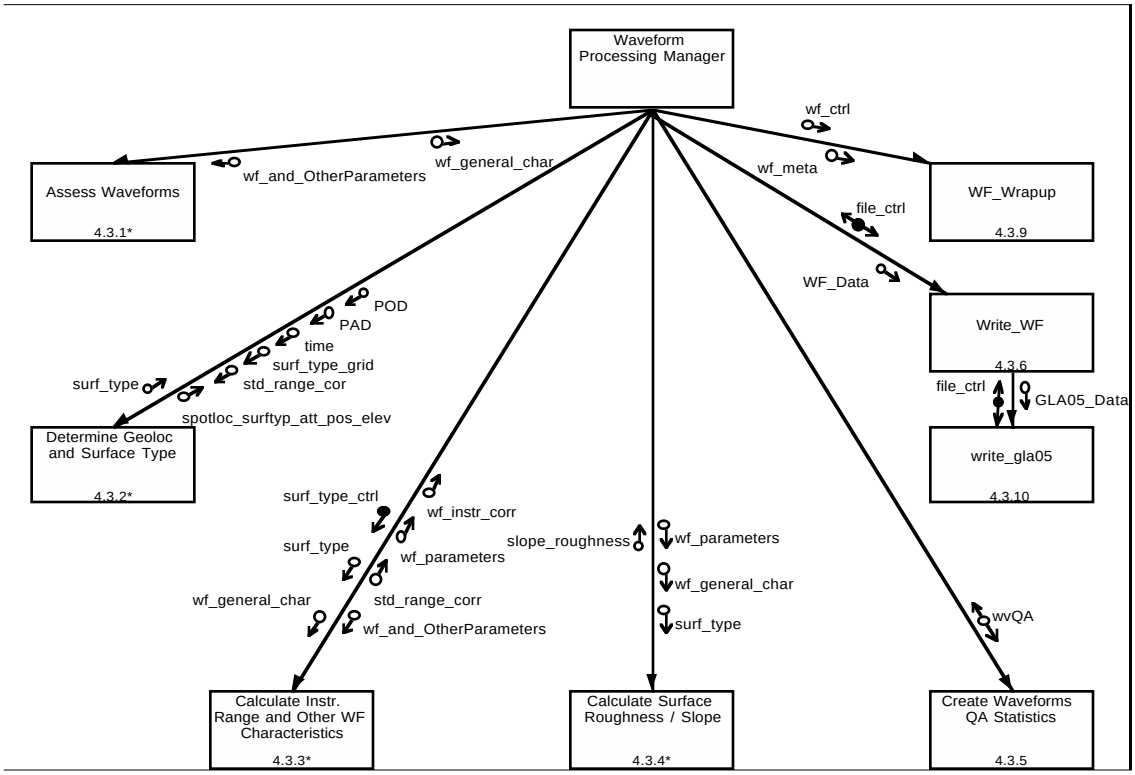

Figure 12-5 Waveform Processing Manager

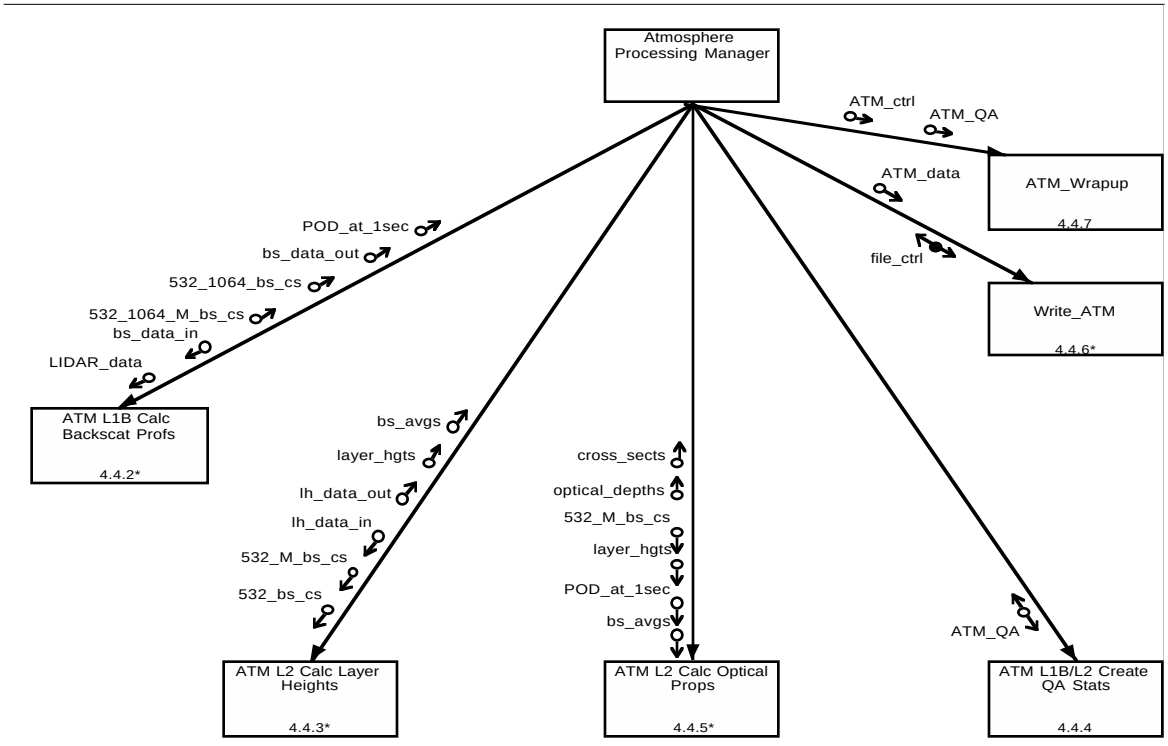

Figure 12-6 Atmosphere Processing Manager

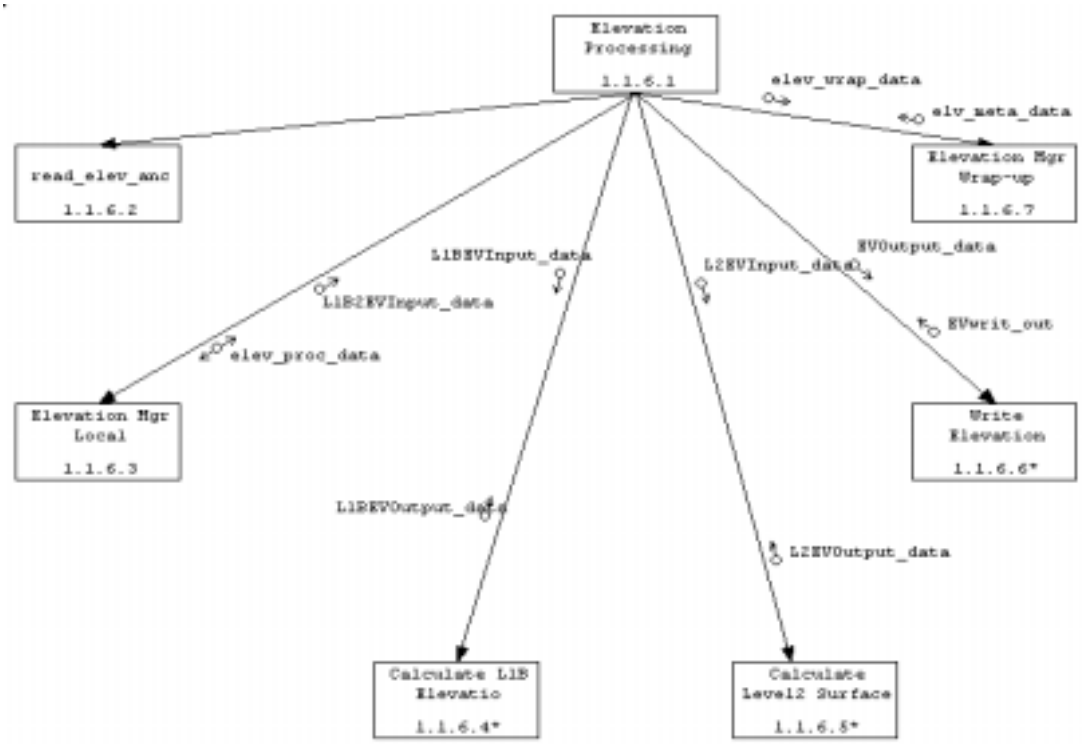

Figure 12-7 Elevation Processing Manager

12.2 Level 1A Computations

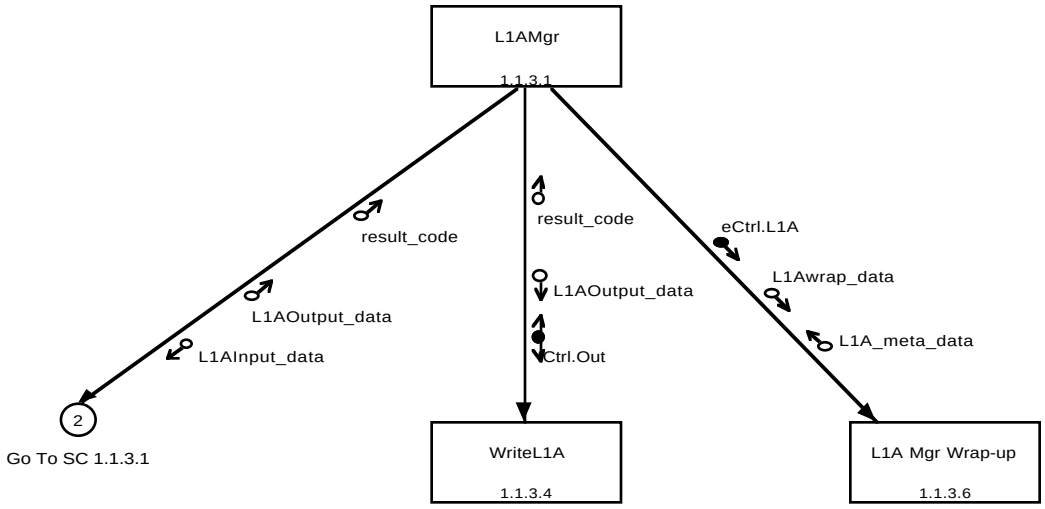

Figure 12-8 Level 1A Computations Manager

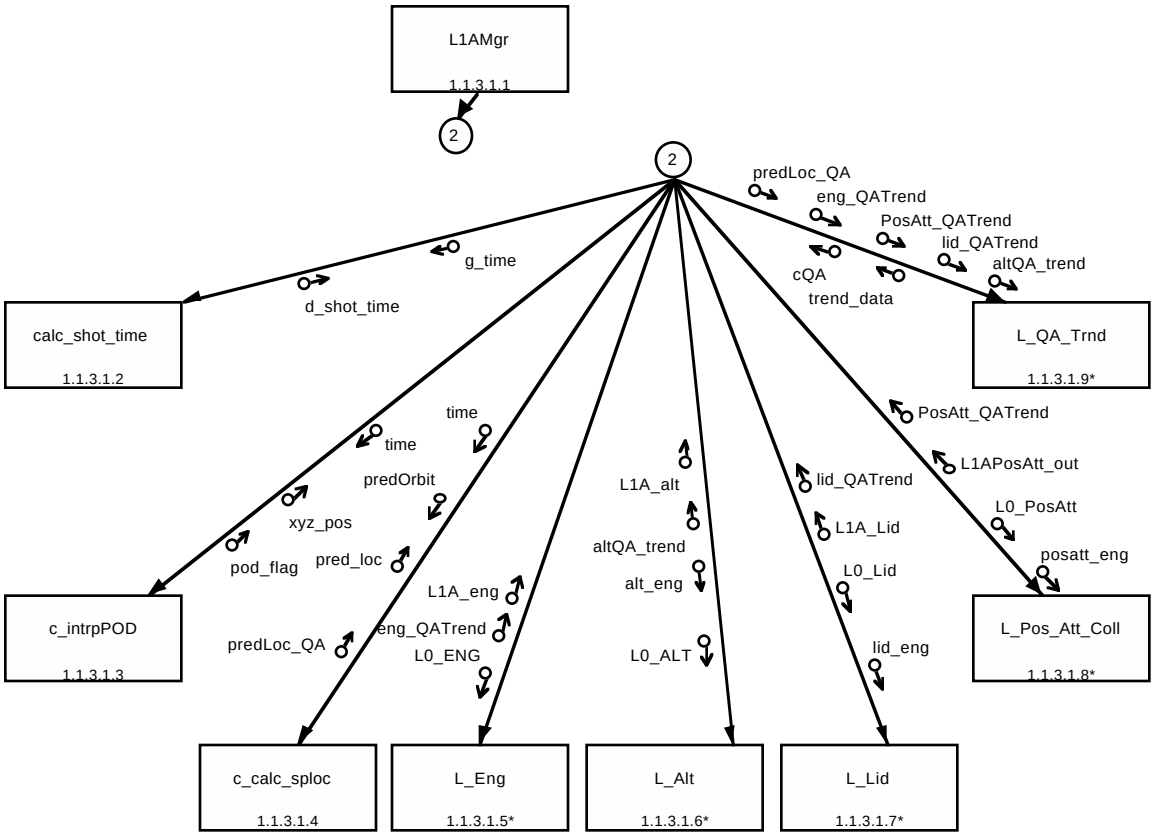

Figure 12-9 Level 1A Computation Manager

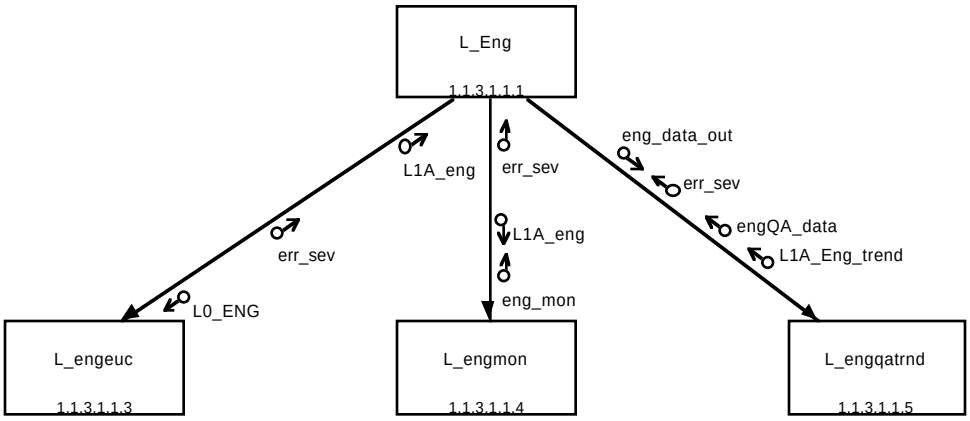

Figure 12-10 Generate Level 1A Engineering Data Product

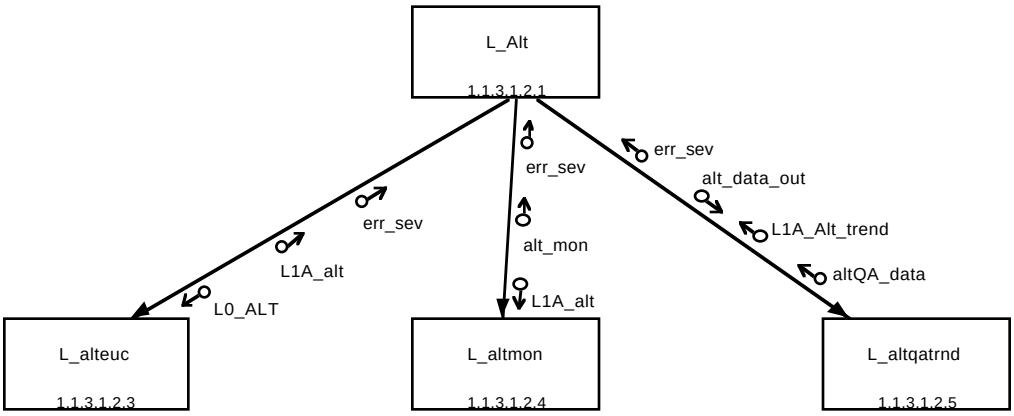

Figure 12-11 Generate Level 1A Altimeter Data Product

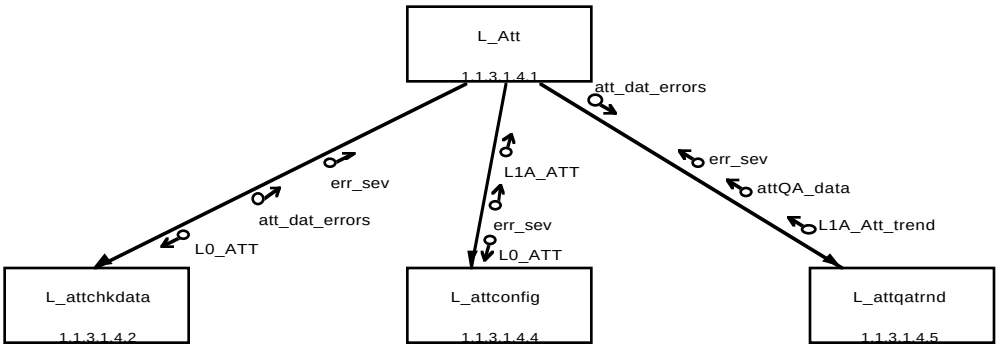

Figure 12-12 Generate Level 1A Attitude Data Product

12.3 Waveforms

12.3.1 Level 1B Waveforms

During normal processing, the waveform processing manager (W_Manager) calls the subprocesses W_Assess, and W_FunctionalFt. Each of these subprocesses stores its QA data. When the subprocesses are finished, W_Manager calls W_CreQAStats, which in turn calls W_GetAsQAStats and W_GetFfQAStats, processes the QA data and returns waveform QA information for the current granule.

For reprocessing, both W_Assess and W_FunctionalFt will be called. The calculation of latitude, longitude and elevation, and the calculation of background noise and standard deviation of noise can be turned off with control settings.

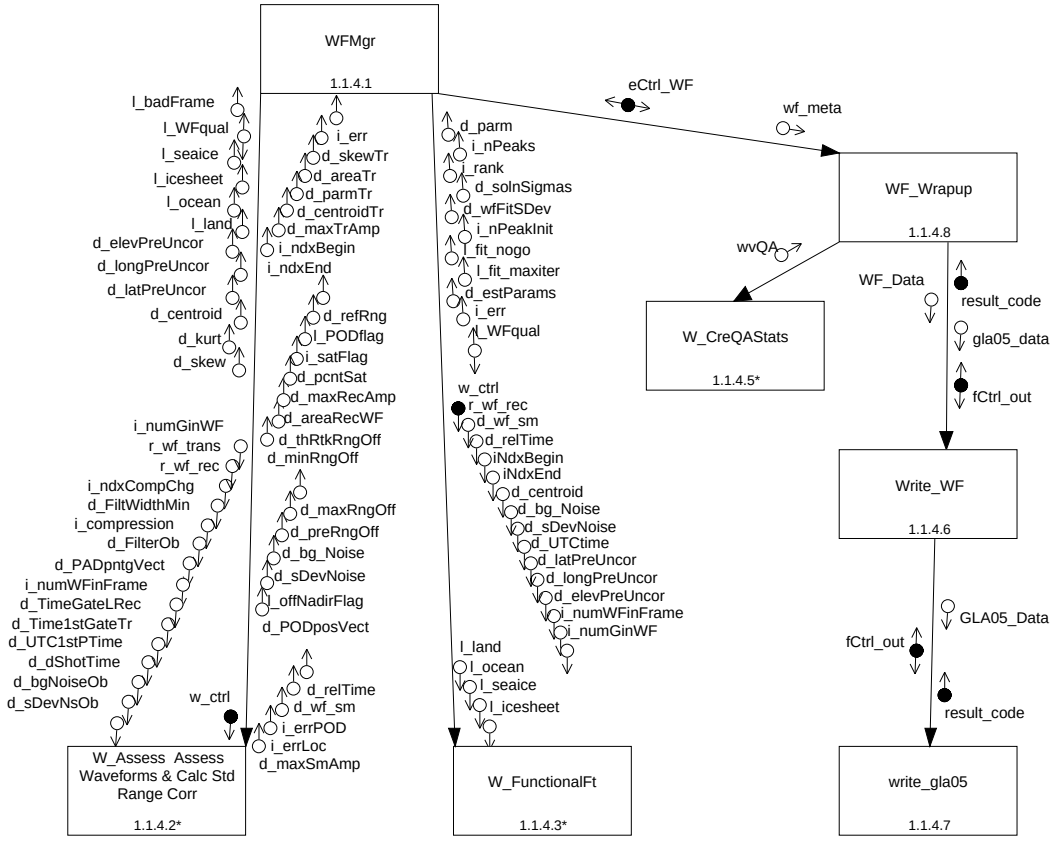

Figure 12-13 Level 1B Waveforms Structure Chart

12.3.2 Assess Waveforms

During processing, W_Assess calls the subprocesses W_CalcRelTime, W_CharTrPulse, W_CalcNoise, W_CalcRefRng, W_SmoothPreRC, W_DetGeo, C_GetRegions, W_Ck4Sat, W_CalcCtMxArAs, and W_CalcThRetrkr. Each of these subprocesses stores its QA data in W_Assess_mod module.

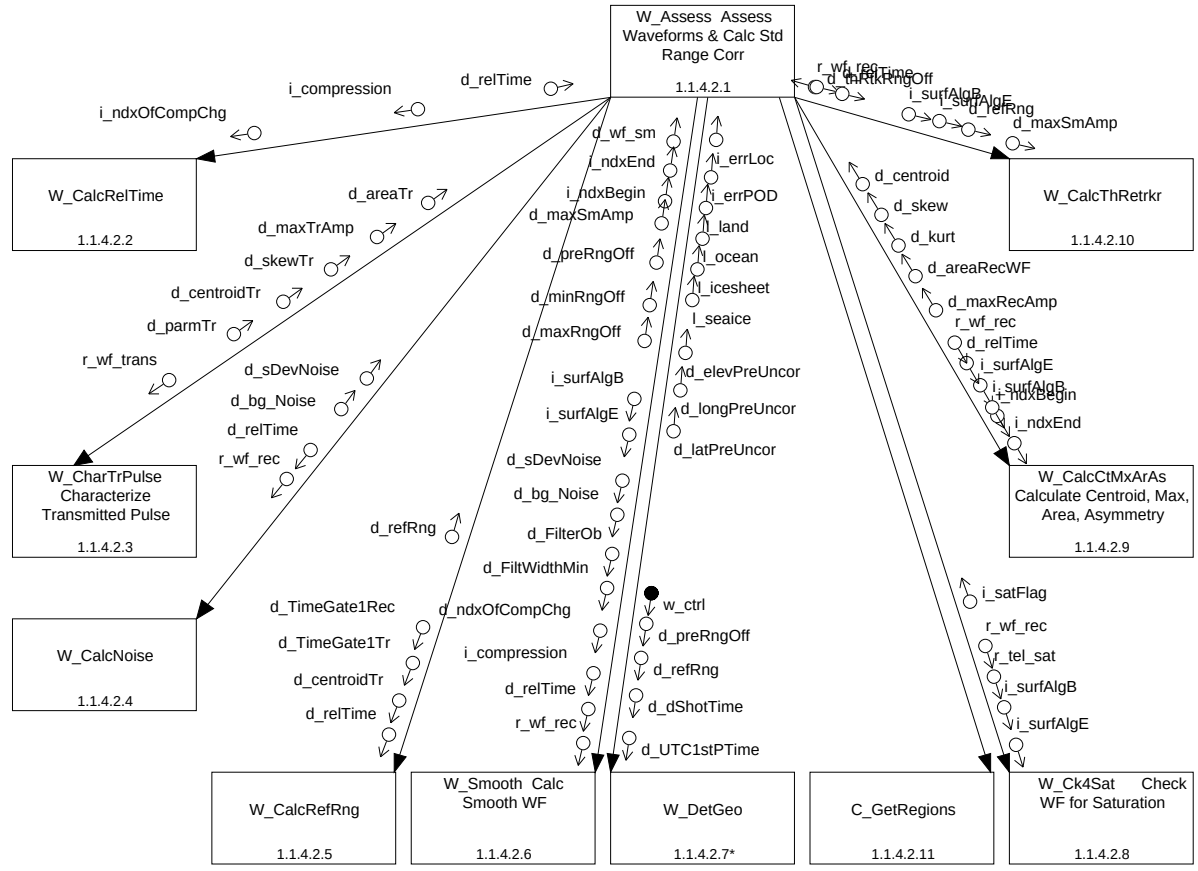

Figure 12-14 Assess Waveforms Structure Chart

12.3.3 Functional Fit

During processing, W_FunctionalFt calls the subprocess W_ParamWithFit. This subprocess and subprocesses called by it store their QA information in the W_FunctionalFt_mod module.

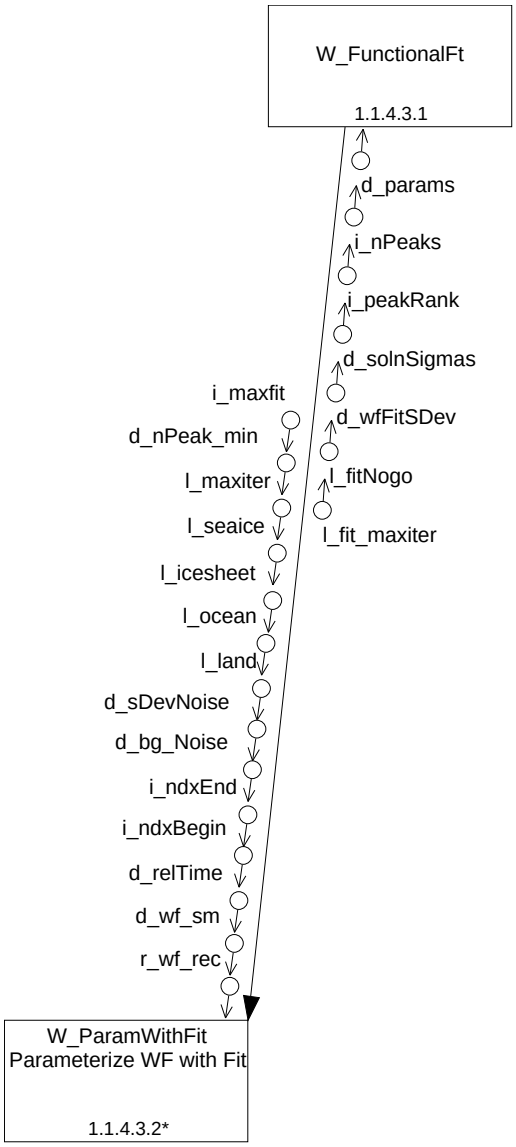

Figure 12-15 Functional Fit

12.4 Level 1B and 2 Atmosphere Computations

The following structure charts illustrate the organization of the atmosphere computations software modules. Modules are called top to bottom and from left to right. Input variables point downwards to the modules that are receiving them while output variables point upwards from the module which created them. Control is not an

argument, but indicates which modules are only selectively called by the atmosphere manager for partial reprocessing.

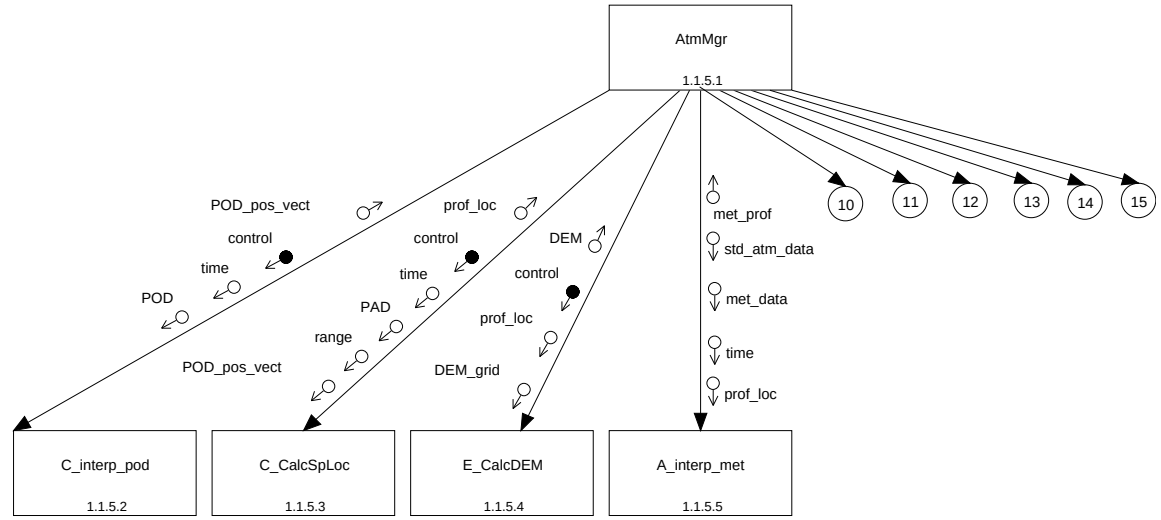

Figure 12-16 Atmosphere Subsystem: Profile Location / Met Modules

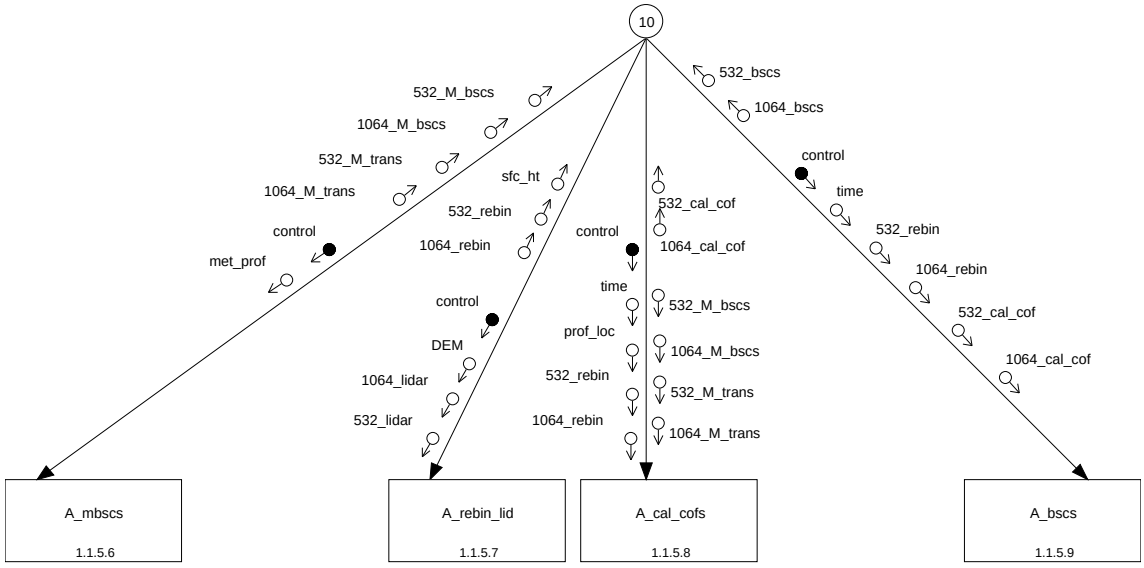

Figure 12-17 Atmosphere Subsystem: Backscatter Modules

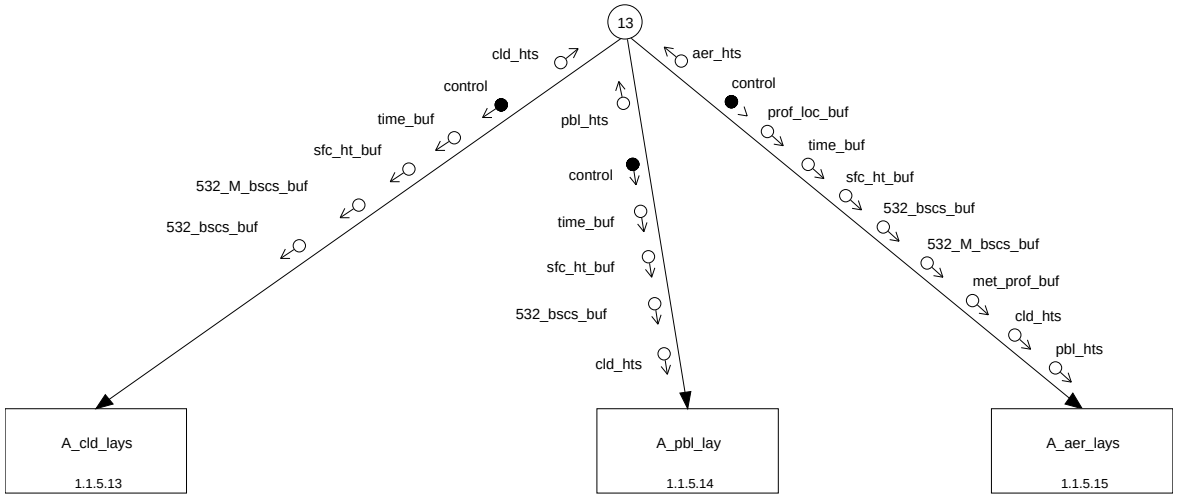

Figure 12-18 Atmosphere Subsystem: Cloud / Aerosol Layer Heights Modules

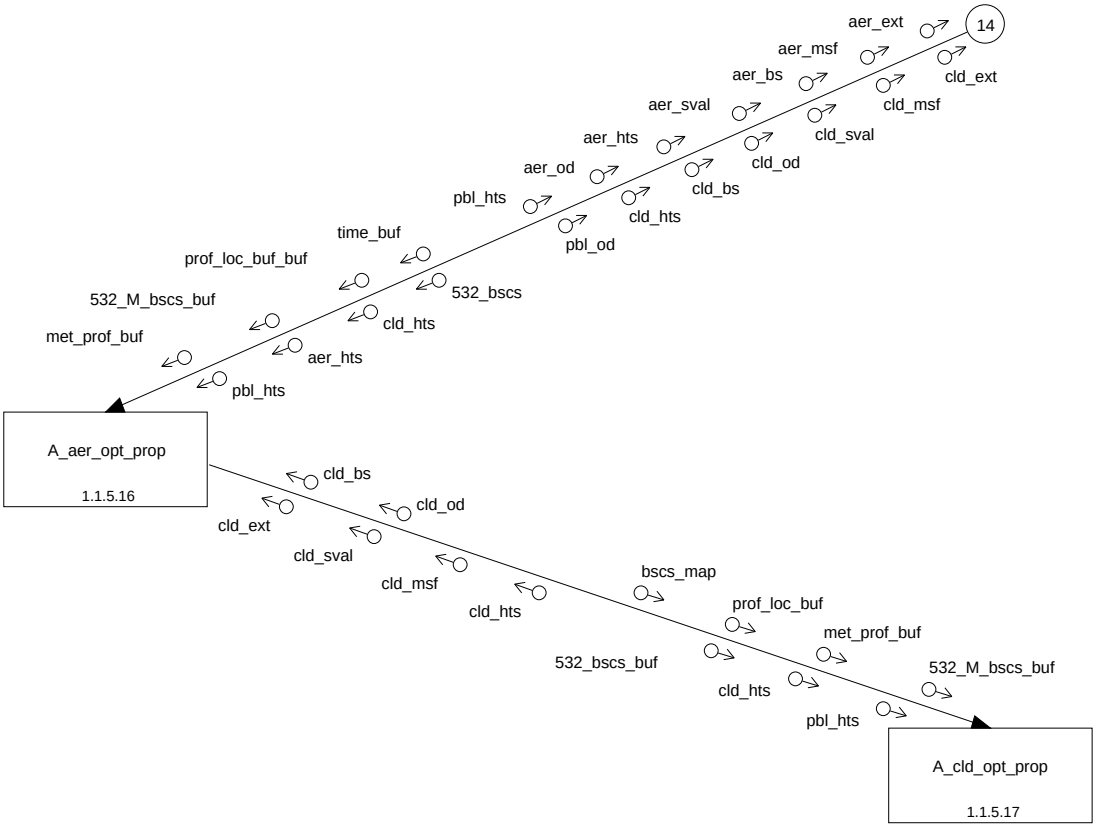

Figure 12-19 Atmosphere Subsystem: Optical Properties Modules

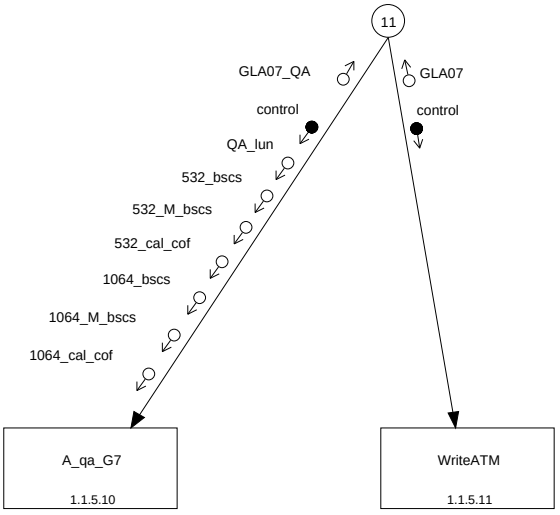

Figure 12-20 Atmosphere Subsystem: QA Statistics Module

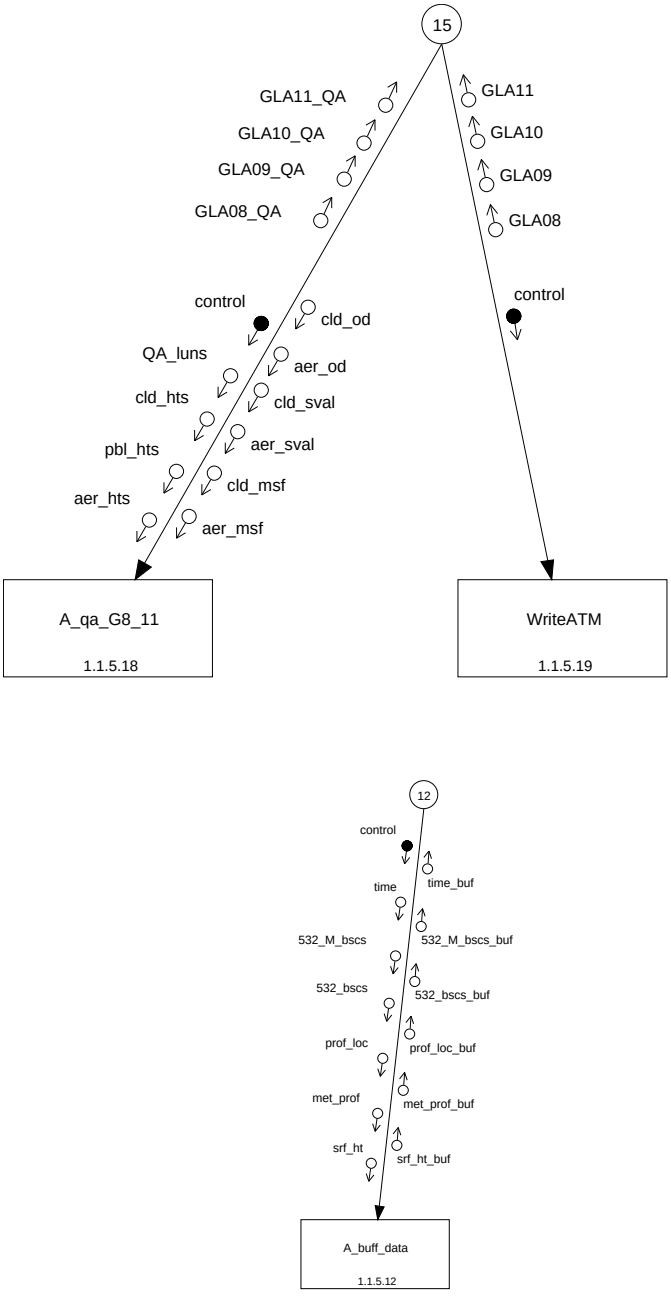

Figure 12-20 Atmosphere Subsystem: QA Statistics Module

12.5 Level 1B and 2 Elevation Structure Charts

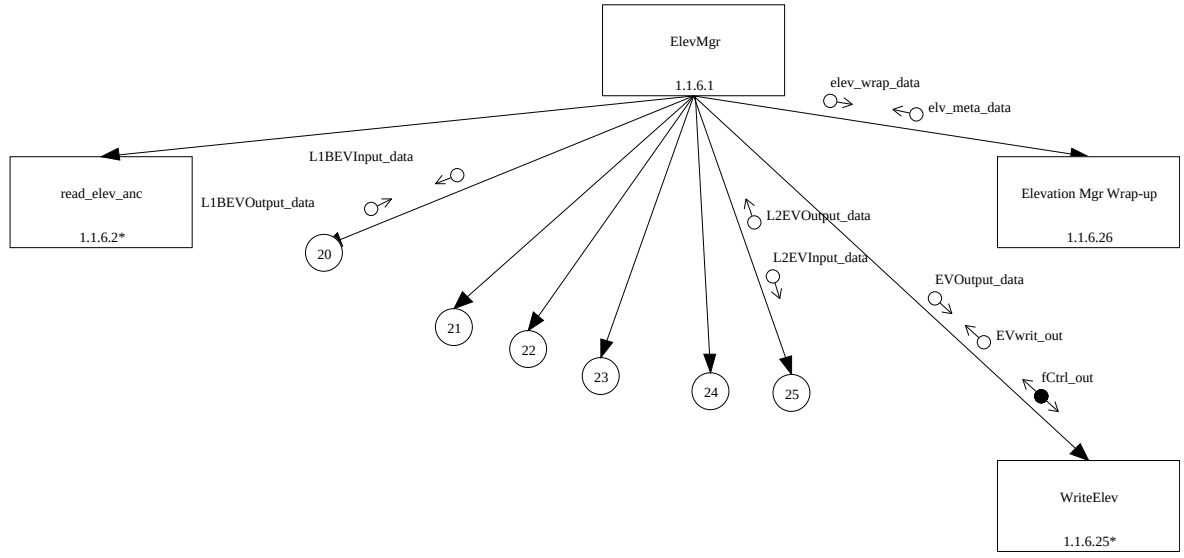

Figure 12-21 Elevation Manager

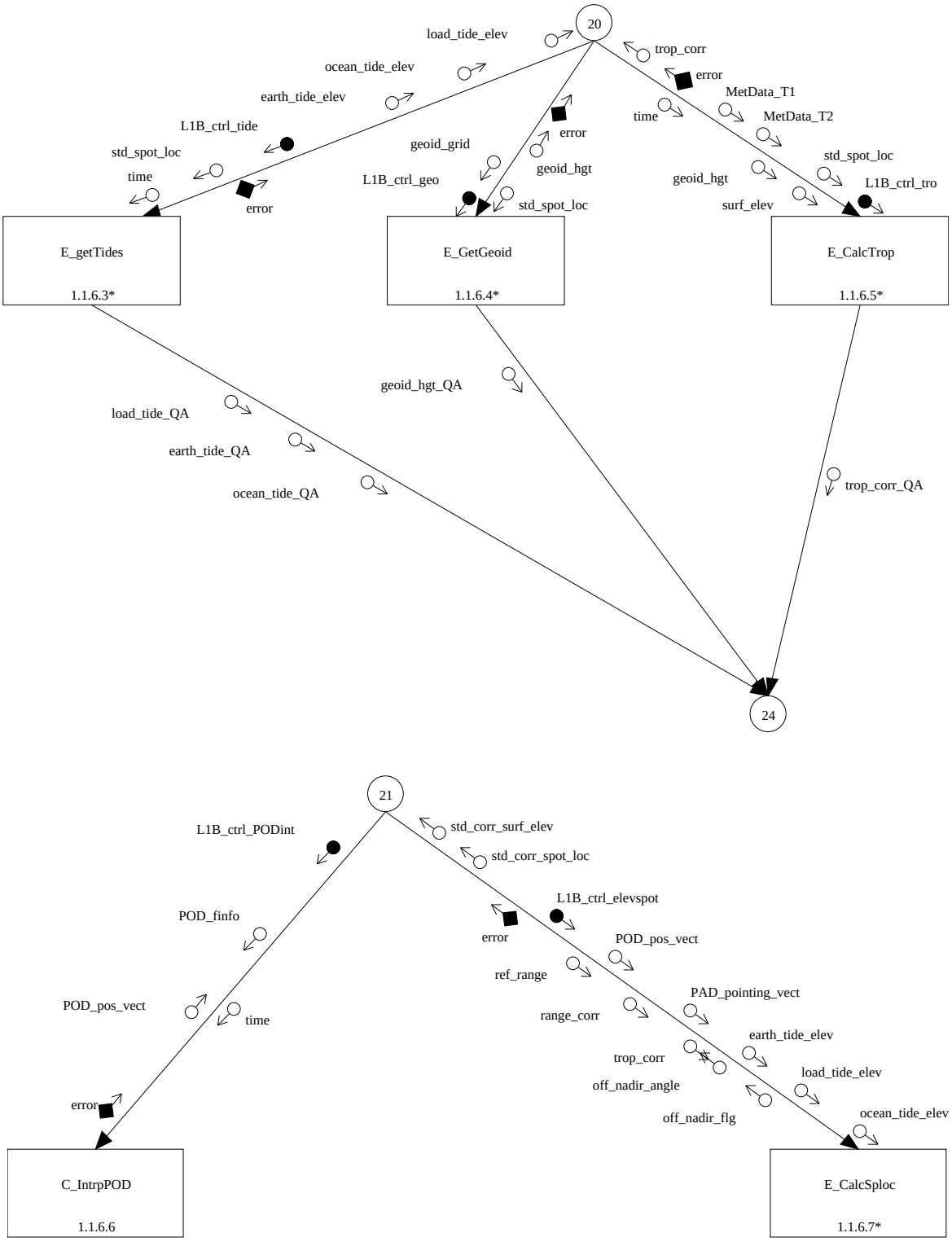

Figure 12-21 Elevation Manager (Continued)

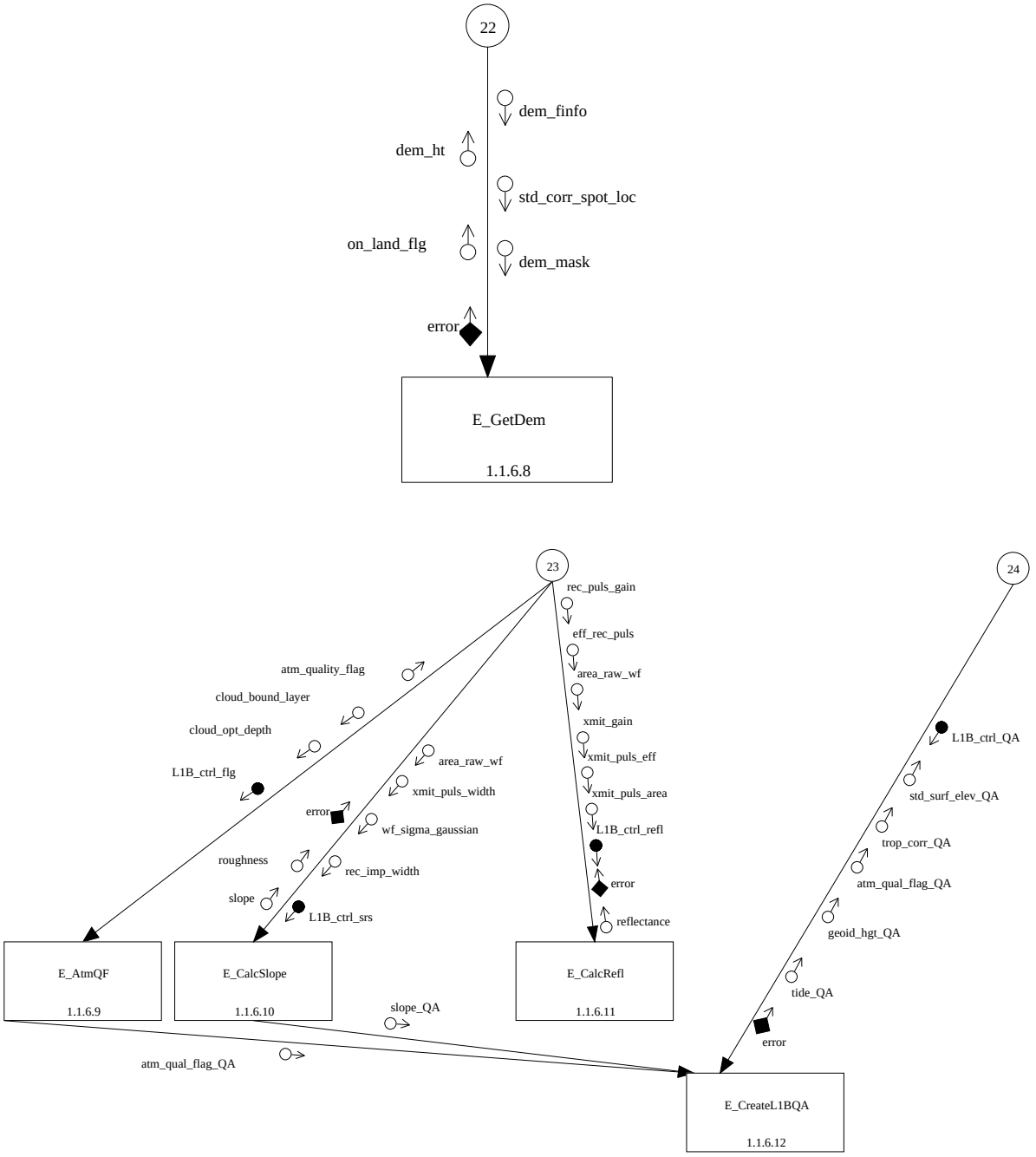

Figure 12-21 Elevation Manager (Continued)

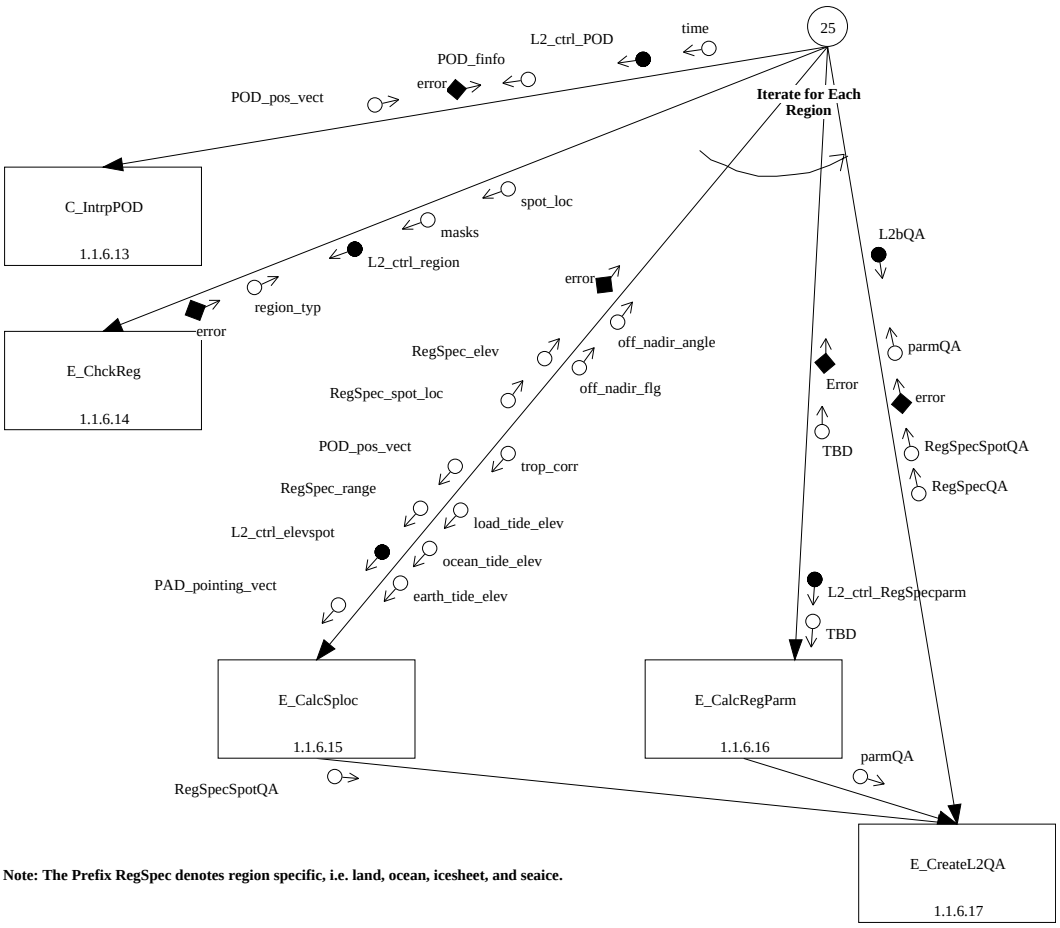

Figure 12-22 Calculate Level 2 Elevations Structure Chart

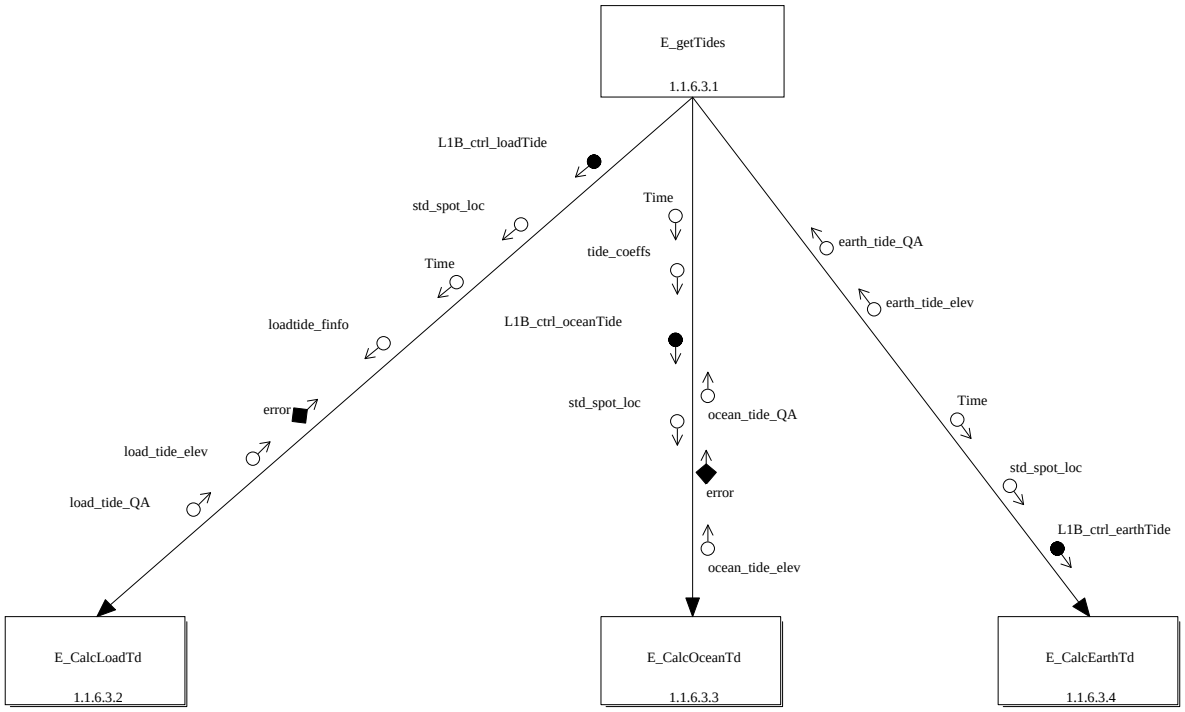

Figure 12-23 Get Tides Structure Chart

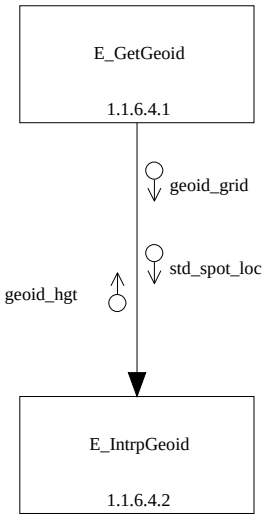

Figure 12-24 Get Geoid Structure Chart

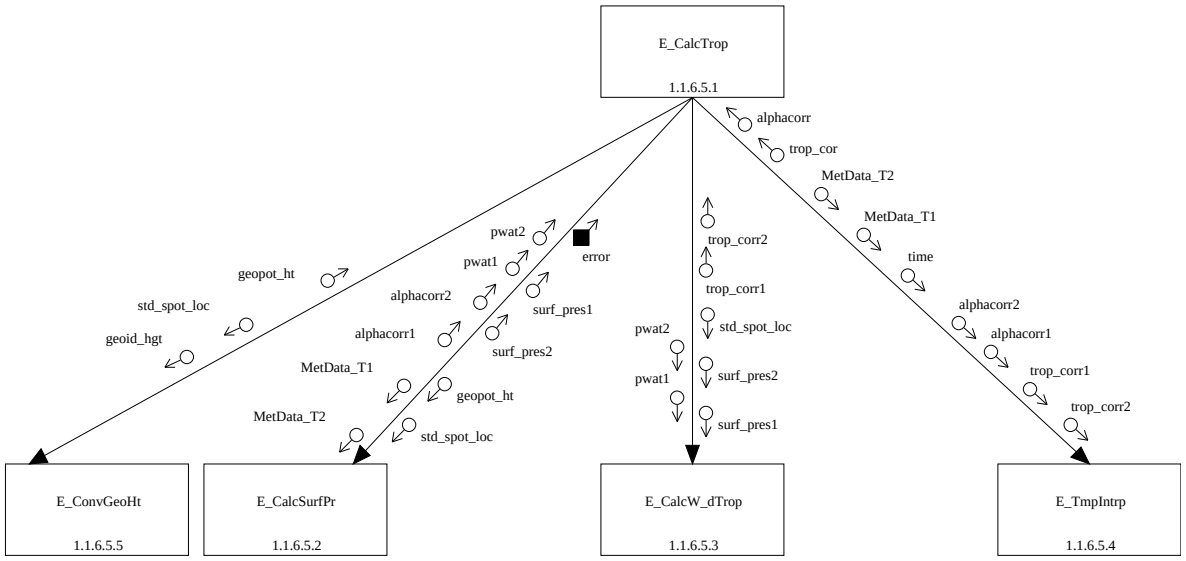

Figure 12-25 Calculate Spot Location Structure Chart

Appendix A
Processing Scenarios

All identified scenarios that will be eventually tested.

Table A-1 Reprocessing Scenarios

Scenario	Input	Output	Dependencies	Processes
End to end	Level 0, ANC data, Cntrl	GLA01-15, Metadata		All
End to end Lidar	Level 0, ANC data (POD, Met, Cal file), Cntrl	GLA02, 7-11, Meta-data		L1A Atm ATBD, L1B Atm ATBD, L2 Atm ATBD, POD interp, Met interp
End to end Altime-ter	Level 0, POD, PAD, Met, Cal file, Cntrl	GLA05,6,12 -15, Meta-data		L1A Altimeter ATBD, L1B Waveform ATBD, L1B Elevation, L2 Elevation, POD, PAD, Geoloc
Level 1A Altimeter	Level 0, Cal file,Cntrl	GLA01, Metadata		L1A Altimeter ATBD
Level 1B Waveform	GLA01, POD, PAD, Cal file, ANC 19, surf_type_grid, Cntrl	GLA05,Met adata		L1B Waveform ATBD, POD, PAD, Geoloc, surf_type interp
Level 1B Elevation	GLA05, GLA09&11 (if avail), tide coeff, geoid, ANC 12, DEM, Met	GLA06, Metadata		Geoid, Tides, Geoloc, Met, DEM interp, Instr Range Cor (5) Reflectance, Atm Flag
Level 2 Elevation	GLA05, GLA06, 4 Masks	GLA12-15, Metadata		Geoloc, Instr Cor Range Region-Specific Parame-ter Calculations
Waveform Algo-rithm changes (standard, ice sheet, sea ice, ocean, land)	GLA01, GLA05, Cal file	GLA05, Metadata	GLA06, GLA12-15 (1 or all)	Specific Wave-form algorithm process, Geolo-cation
Replace POD and/ or PAD on GLA05	GLA05, POD and/or POD	GLA05, Metadata		POD and/or PAD, Geolocation

Table A-1 Reprocessing Scenarios (Continued)

Scenario	Input	Output	Dependencies	Processes
Replace PAD and/ or POD on GLA06	GLA06, PAD and/or POD	GLA06, Metadata	GLA12-15	PAD and/or POD, Geolocation
Met changes, redo Met Cor	GLA06, GLA12-15, Met file	GLA06, GLA12-15, Metadata		Met Interpola-tion, Geolocation
Tides Change, redo tide cor	GLA06, GLA12-15, tide coeff	GLA06, GLA12-15, Metadata		Tide algorithms, Geolocation
Geoid changes	GLA06, GlA12-15, Geoid	GLA06, GLA12-15, Metadata		Geoid
Standard Instr Cor Changes	GLA05, GLA06, GLA12-15	GLA06, GLA12-15, Metadata		Standard Instr Cor Algorithm, Geolocation
Region Spec Instr Cor Changes	GLA05, GLA06, GLA12-15	GLA06, GLA12-15, Metadata		Region Specific Instr Cor Algo-rithm
Reflectance Algo-rithm changes	GLA05, GLA06, GLA12-15	GLA06, GLA12-15, Metadata		Reflectance ATBD
Change GLA06 based on WF Algo-rithm changing for GLA05	GLA05, GLA06, GLA12-15	GLA06, GLA12-15, Metadata		Range Instr Cor Calculation, Geolocation
Replace PAD and/ or POD on GLA12-15	GLA12-15, PAD and/or POD	GLA12-15, Metadata		POD and or PAD, Geolocation
Creation of GLA07 BackScatter Pro-files	GLA02, Met, POD, 400 sec avg file	GLA07, Metadata		Interp POD,Interp Met, Molec Back-Scat Profiles, Calib Coeff, 1064 BackScat Pro-files, 532 Back-Scat Profiles
Creation of GLA08 Aerosol Layers	GLA07, Constants, GLA09	GLA08		1 and 4 sec BackScat aver-ages, PBL/Aero-sol <20 km layers, 20-40 km aerosol layers

Table A-1 Reprocessing Scenarios (Continued)

Scenario	Input	Output	Dependencies	Processes
Creation of GLA09 Cloud Layers	GLA07, Constants	GLA09		1 and 4 sec BackScat averages, Cloud Layers
Creation of GLA10 Cross Section Profiles and Creation of GLA11 Optical Depths	GLA07, GLA08, GLA09, Constants	GLA10, GLA11		Cloud Optical Properties, Aerosol Optical Properties, 1 and 4 sec BackScat averages

Appendix B
Flow Charts

B.1 GSAS Overview

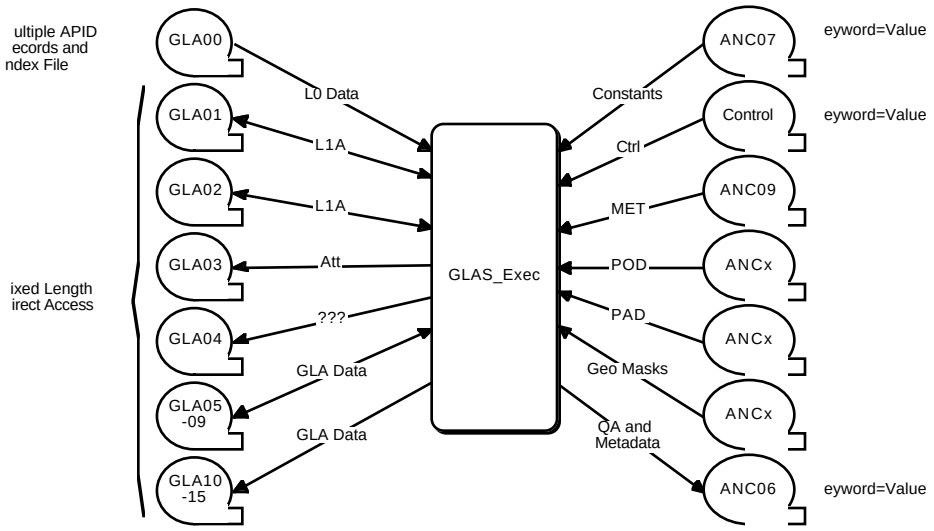

Figure B-1 GSAS Overview

B.2 GLAS_Exec

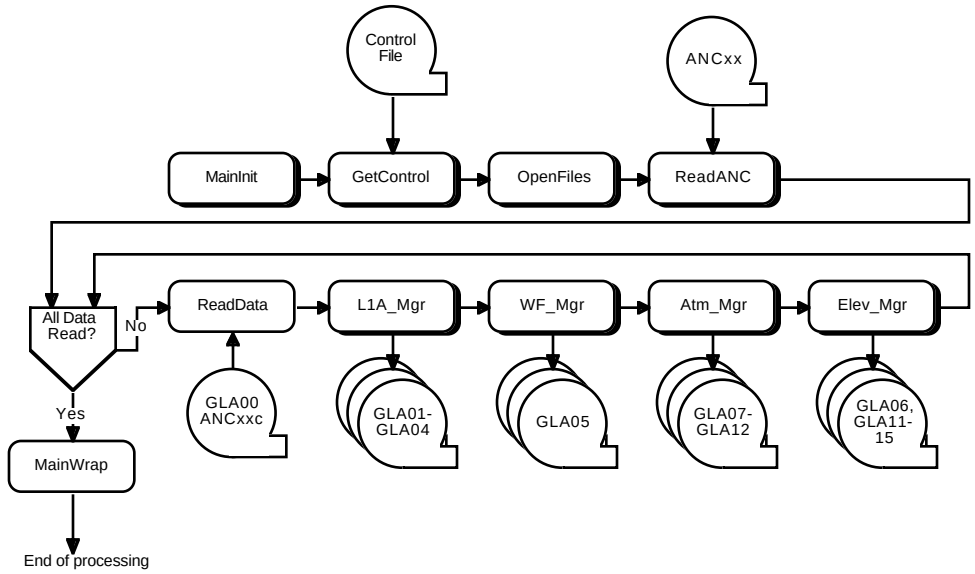

Figure B-2 GLAS_Exec Top-Level

B.3 L1A Manager

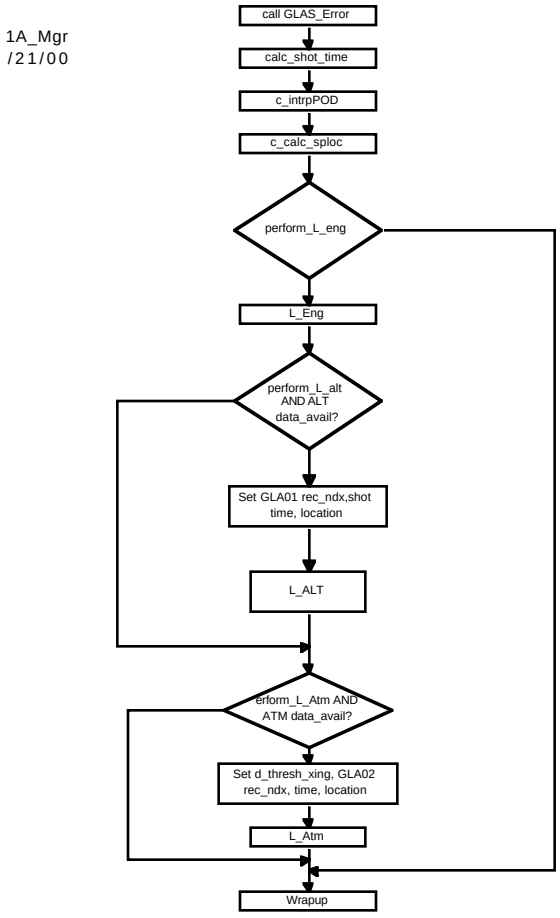

Figure B-3 L1A Manager

B.4 Waveforms Manager

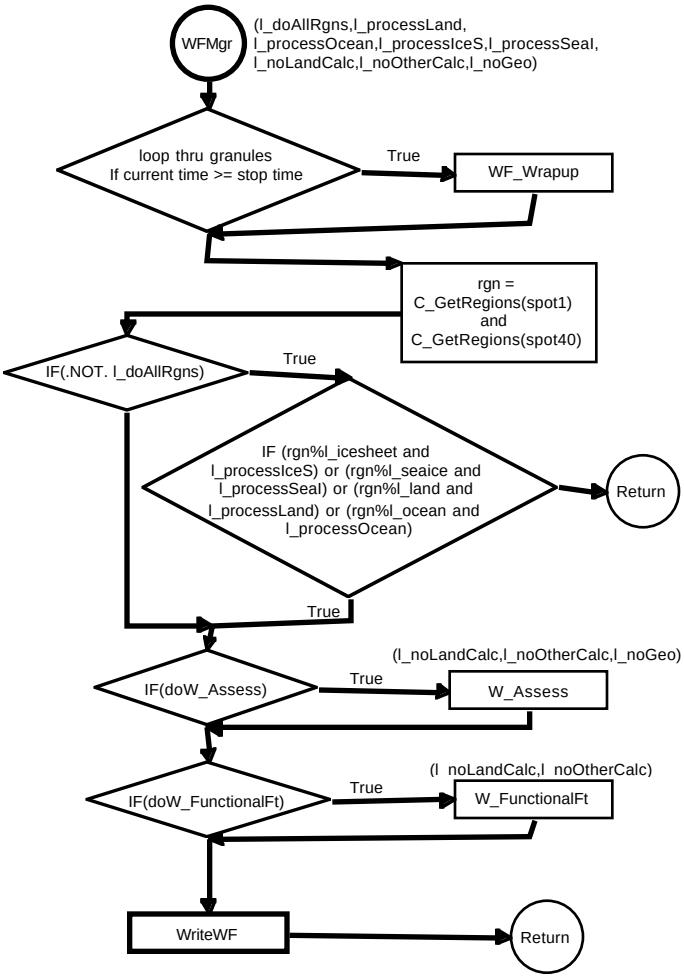

Figure B-4 Waveforms Manager

B.5 Atmosphere Manager

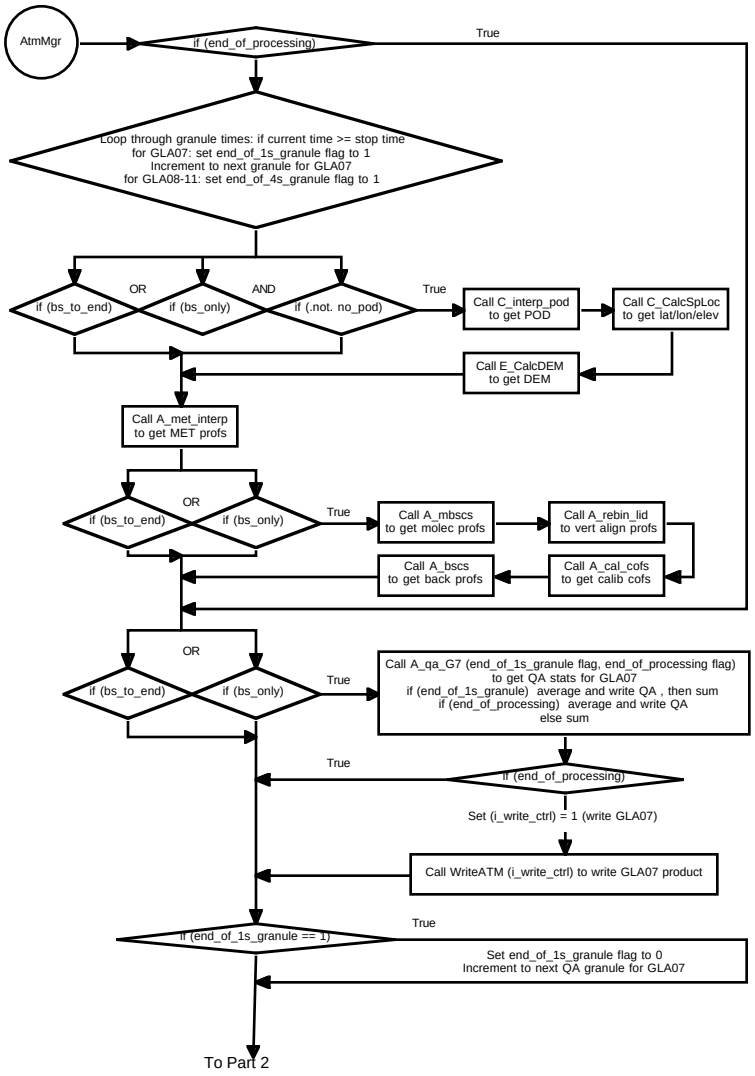

Figure B-5 Atmoshere Manager-Part 1

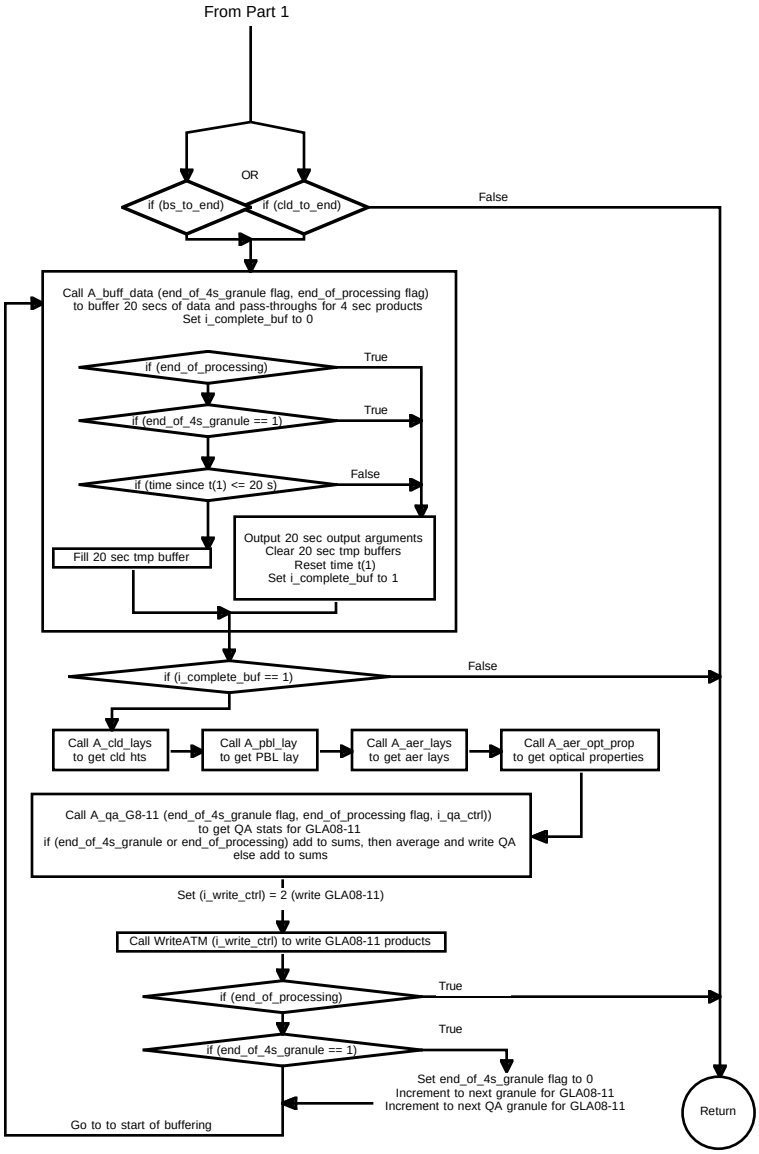

Figure B-6 Atmoshere Manager-Part 21

B.6 Elevations Manager

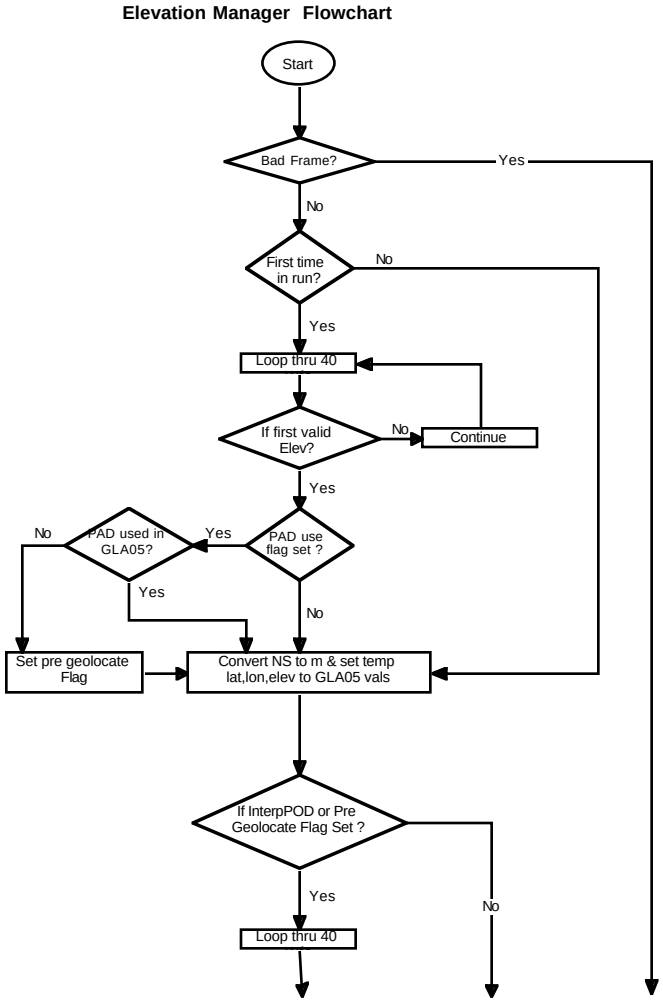

Figure B-7 Elevations Manager - Part 1

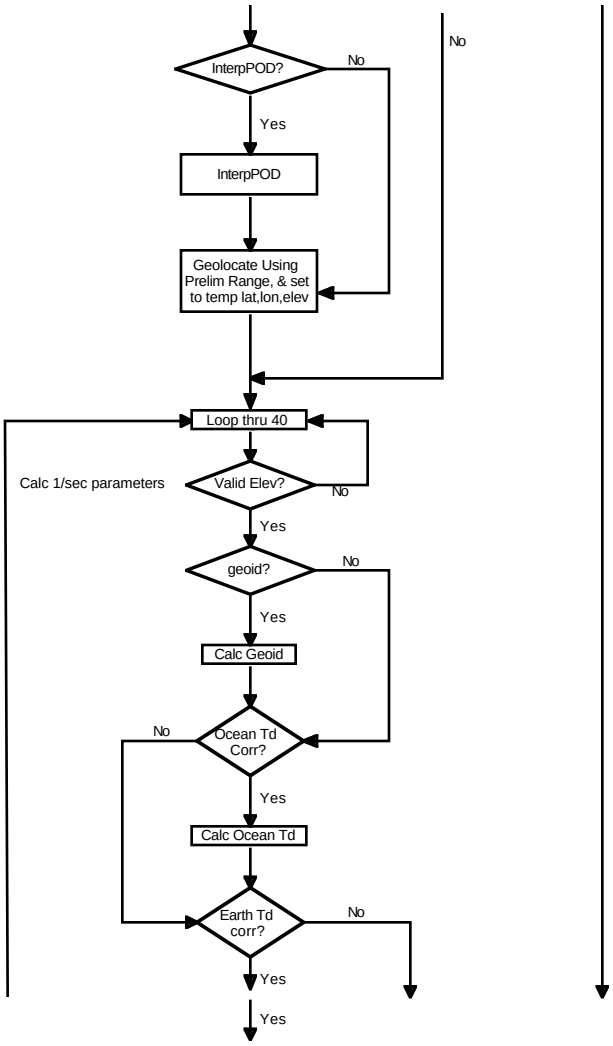

Figure B-8 Elevations Manager - Part 2

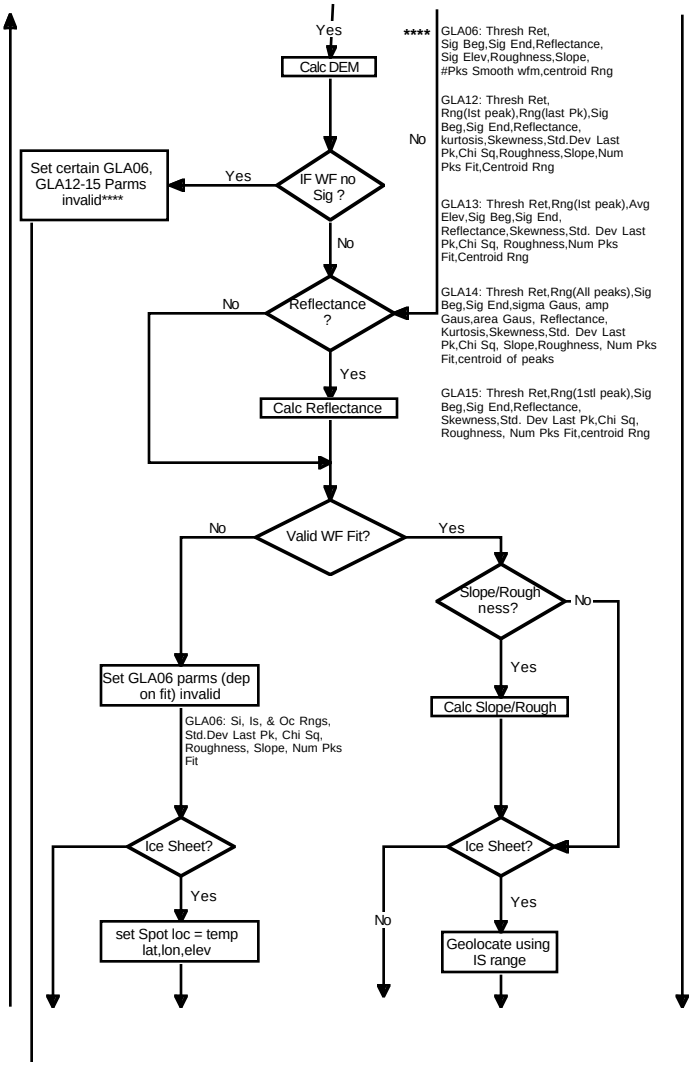

Figure B-9 Elevations Manager - Part 3

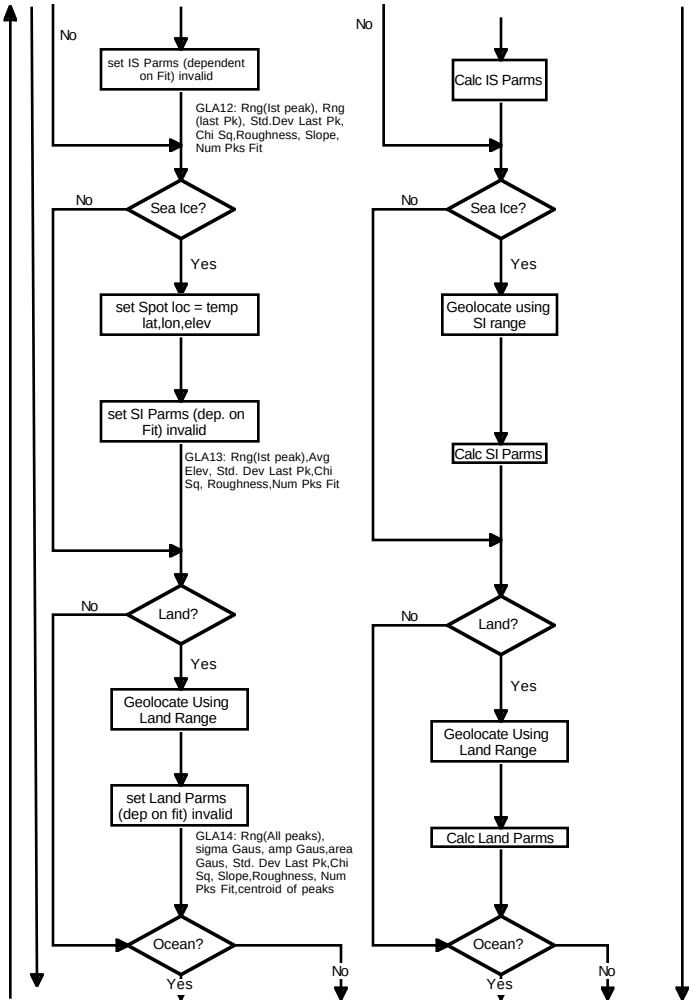

Figure B-10 Elevations Manager - Part 4

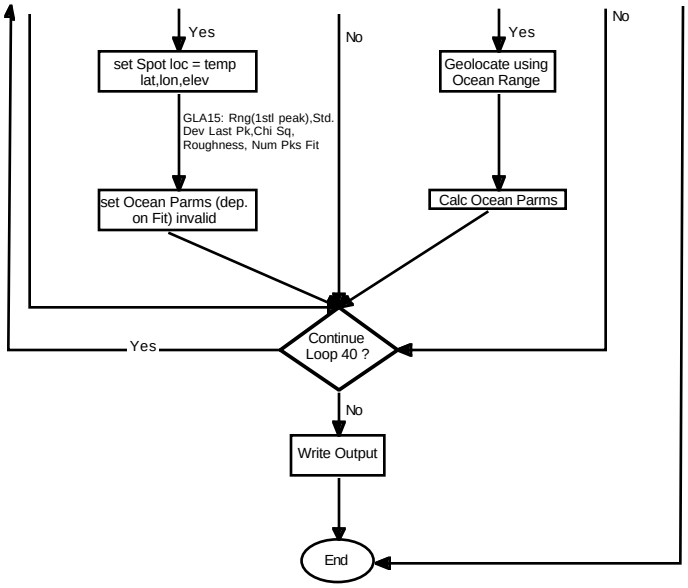

Figure B-11 Elevations Manager - Part 5

Appendix C
Control File Format

The GLAS_Exec Control File is generated either by hand or automatically by the SDMS. The format is standard GSAS “keyword=value”. The format should be consistent with the following conventions:

- Keywords and values (except for filenames) are not case sensitive.
- Comments must be preceded by the "#" sign.
- Blank lines are allowed, but not recommended.
- Sections have been slightly modified to coincide with SDMS sections. The start of a section is delimited by an “=” in the first character position. The value after the “=” defines the name of the section.

Table C-1 lists acceptable keyword / value combinations for the GLAS_Exec section of the control file.

Table C-1 Control File Keywords and Values

Keyword	Acceptable Values
TEMPLATE	(filename:string)
EXEC_KEY	(sequence:integer)
DATE_GENERATED	(date:string)
OPERATOR	(userid:string)
INPUTFILE	(filename:string) (starttime:float) (stoptime:float)
OUTPUTFILE	(filename:string) (starttime:float) (stoptime:float)
CYCLE	(cycle:integer) (starttime:float) (stoptime:float)
REV	(rev:integer) (starttime:float) (stoptime:float)
SURFACE_TYPE	ALL
SURFACE_TYPE	Land
SURFACE_TYPE	Ocean
SURFACE_TYPE	Sea_Ice
SURFACE_TYPE	Ice_Sheet
L1A_PROCESS	ALL
L1A_PROCESS	NONE

Table C-1 Control File Keywords and Values (Continued)

Keyword	Acceptable Values
L1A_PROCESS	L_Alt
L1A_PROCESS	L_Atm
L1A_PROCESS	L_Att
L1A_PROCESS	L_Eng
WAVEFORM_PROCESS	ALL
WAVEFORM_PROCESS	NONE
WAVEFORM_PROCESS	W_Assess
WAVEFORM_PROCESS	W_Fuctional_Ft
WAVEFORM_PROCESS	W_CreaQAStats
WAVEFORM_OPTION	W_doAllRgns
WAVEFORM_OPTION	W_processLand
WAVEFORM_OPTION	W_processOcean
WAVEFORM_OPTION	W_processIceS
WAVEFORM_OPTION	W_processSeaI
WAVEFORM_OPTION	W_noLandCalc
WAVEFORM_OPTION	W_noOtherCalc
WAVEFORM_OPTION	W_noGeo
ATMOSPHERE_PROCESS	ALL
ATMOSPHERE_PROCESS	NONE
ATMOSPHERE_PROCESS	A_bs_to_end
ATMOSPHERE_PROCESS	A_bs_only
ATMOSPHERE_PROCESS	A_cld_to_end
ATMOSPHERE_PROCESS	A_no_pod
ELEVATION_PROCESS	ALL
ELEVATION_PROCESS	NONE
ELEVATION_PROCESS	E_CalcLoadTD

Table C-1 Control File Keywords and Values (Continued)

Keyword	Acceptable Values
ELEVATION_PROCESS	E_CalcOceanTD
ELEVATION_PROCESS	E_CalcEarthTD
ELEVATION_PROCESS	E_CalcPoleTD
ELEVATION_PROCESS	E_GetGeoid
ELEVATION_PROCESS	E_CalcTrop
ELEVATION_PROCESS	E_IntrpPOD
ELEVATION_PROCESS	E_CalcStdIR
ELEVATION_PROCESS	E_CalcLdIR
ELEVATION_PROCESS	E_CalcOcIR
ELEVATION_PROCESS	E_CalcSiIR
ELEVATION_PROCESS	E_CalcIsIR
ELEVATION_PROCESS	E_CalcSpLoc
ELEVATION_PROCESS	E_AtmQF
ELEVATION_PROCESS	E_CalcSlope
ELEVATION_PROCESS	E_CalcRefl
ELEVATION_PROCESS	E_ChckReg
ELEVATION_PROCESS	E_CalcRegRng
ELEVATION_PROCESS	E_CalcRegParm
ELEVATION_PROCESS	E_CalcDEM

Appendix D
Makefiles and Libraries

Developers are “strongly” encouraged to use standard GSAS libraries and makefiles. GSAS libraries leverage existing code to speed development and ease maintenance. Makefiles ensure common compiler flags and allow developers to deliver their software as part of a general GSAS delivery.

D.1 Library Compilation

Note: This documentation is specific to the GLAS development environment. It assumes that the core directory of the GLAS software is located at “/glas/vob”.

D.1.1 To create the libraries:

```
cd /glas/vob/src
make libs
```

D.1.2 To create GLAS_Exec

```
cd /glas/vob/src
make libs; make GLAS_Exec
```

D.1.3 To create prod_readers

cd /glas/vob/src
make libs; make prod_readers

D.1.4 To recompile a library in debug mode

```
cd /glas/vob/src/{library_path}
make debug
```

D.2 Using Libraries

This section details the use of libraries, both at development and run time stages.

D.2.1 Development

To use a library, you need to include the path and the library name in your Makefile. The following example shows how to use the platform_lib (which is stored in the /glas/vob/src/lib directory) to compile a test program:

```
f90 test.f90 -L/glas/vob/src/lib -lplatform -otest
```

The next example show how to use the file and anc libraries as well. (A side note: By unix convention the full filenames are libplatform.sl, libfile.sl, and libanc.sl, however when they are specified with the -l argument on the compile line, the “lib” and “.sl” parts are dropped).

```
f90 test.f90 -L/glas/vob/src/lib -lplatform -lanc -lfile –otest
```

ORDER IS IMPORTANT. See Foundation Libraries section of this document to verify that the libraries are specified in the correct order on the compile line.

D.2.2 Runtime

GSAS libraries are dynamically-linked shared libraries. What this means is that the libraries are not statically linked with executables, but dynamically linked on demand at runtime. With this in mind, it is important that the executable be able to determine the location of the libraries at runtime. During compilation, the location of the libraries is stored in the executable code. If the executable is moved, and the location is relative, the libraries will not be found upon execution. In this case, a developer should use the following procedure to allow executables to link to dynamic libraries, no matter their location.

```
chatr +s enable <executable_name>
setenv SHLIB_PATH <pathname to libraries>
```

This procedure tells the executable to use the SHLIB_PATH environmental variable to find its libraries, then sets that variable to the path of the shared libraries.

The other way of handling this is to link the libraries into the current directory. The executable is set to look in the current directories first for its libraries.

D.3 Some Development Hints

- If you want to use the GLAS libraries, simply compile them (as above) and include the appropriate lines in your makefile (again, as above).
- As long as you model the Makefile for your executable after that of GLAS_Exec, you will be using shared libraries and will not need to recompile your executable after recompiling a library.
- If you would like to debug the routines in a specific library, cd to that directory and do a make clean; make debug. Next time you run your executable (you don’t have to recompile it), it will run with the debug version of the library.
- Using the –g and +check=all flags (included with make debug) is a good idea during testing.
- If you want to get fancy and create a custom makefile for a special purpose, simply use another name for the makefile and use make –f mymakefile.

D.4 Makefile Details

This is an attempt to explain how GLAS makefiles work. This assumes the reader is somewhat familiar with the GLAS VOB layout.

D.5 Types of Makefiles

There are different types of makefiles. This section identifies each.

D.5.1 The Main Makefile

This makefile is located at /glas/vob/src/Makefile. It is the main makefile which will recursively build all GSAS deliverable software. There are options to:

- build all deliverable GLAS Libraries (make libs)
- build all deliverable GLAS binaries (make progs)
- build all deliverable Libraries and binaries (make all) -the default
- build all deliverable Libraries and binaries in debug mode (make debug)
- build all deliverable Libraries and binaries in optimization mode (make fast)
- clean up all object code and module files (make clean)

D.5.2 Library Makefiles

These makefiles are located at src/common_libs/*/Makefile. There are options to:

- Compile library source and install library (make)
- Compile library source in debug mode and install library (make debug)
- Compile library source in optimization mode and install library (make fast)

The libraries are derived objects and installed into /glas/vob/src/lib

D.5.3 Subsystem Makefiles

These makefiles are located at /glas/vob/src/*_lib/Makefile (where * = l1a, atm, elev, wf)

- Compile library source and install library (make)
- Compile library source in debug mode and install library (make debug)
- Compile library source in optimization mode and install library (make fast)

The libraries are derived objects and installed into /glas/vob/src/lib. When installed, the libraries are stored in /glas/vob/lib.

D.5.4 Exec makefiles

These makefiles are located in the diectory of each delivered executable: src/GLAS_Exec/Makefile, src/prod_readers/Makefile and, src/L0_Proc/Makefile. There are options to:

- Compile binary source (make)
- Compile binary source in debug mode (make debug)
- Compile binary source in optimization mode (make fast)

The executables are derived objects and installed into /glas/vob/bin.

D.6 A Sample Heavily-Commented Makefile

```
# NAME:      Makefile
#
# FUNCTION: Makefile for GLAS_Exec
#
# FILES ACCESSED: See TARGETS definition
#
# COMMENTS: None.
#
# HISTORY:
#
#   1998 December  18, JLee, Initial Version
#   1999 January   14, JLee, Ported to HP
#   1999 October   18, JLee  Removed default DEBUG, removed recursion
#   1999 October   24, JLee  Set the bit to do SHLIB_PATH
#
#----- Set filepaths
#
# PATHLVL is the path you use to get to /glas/vob/src, but it should
#         be a relative path so that we can compile outside the VOB.
#
PATHLVL=..
#
#  UTILDIR is where the GLAS makefile includes can be found. These files
#          contain settings specific to GLAS Makefiles.
#          /glas/vob/cc_util is the actual path.
#
UTILDIR=$(PATHLVL)/../cc_util
#
# Include Standard GLAS Definitions
#
include $(UTILDIR)/make_defs.$(BRAND)
include $(UTILDIR)/make_defs.incl
#
# Define libraries we will need. They are located in /glas/vob/src/lib.
# This path is pre-defined (relatively) in the GLAS include files.
# The actual filename for –lwf is libwf.sl, -file is libfile.sl
#
LIBS= -ll1a -latm -lwf -lelev -lprod -lfile -ltime -lanc -lcntrl \
-lerr –lplatform
#
# Define the Production directory where we will copy the binary upon
# creating a production build
#
PRODDIR=$(PATHLVL/../bin)
#
# Define the target binary
#
TARGET=GLAS_Exec
#
# Define the objects will are needed by the Target
#
OBJECTS= \
       CntlDefs_mod.o fCntl_mod.o eCntl_mod.o \
       MainInit_mod.o ReadData_mod.o GetControl_mod.o \
       CloseFiles_mod.o OpenFiles_mod.o WriteL1A_mod.o WriteWF_mod.o \
       WriteAtm_mod.o WriteElev_mod.o MainWrap_mod.o \
```

```
       ReadAnc_mod.o L1AMgr_mod.o ElevMgr_mod.o WFMgr_mod.o AtmMgr_mod.o \
       vers_exec_mod.o GLAS_Exec.o
#
# Custom Rules
#
gF90_AUX_FLAGS=
#
# Make our Target by default
#
all: $(TARGET)
#
# TARGET, LIBS and OBJECTS are defined in this makefile.
# LINK_EXE.f90 and FFLAGS are defined in the GLAS includes.
# chart +s enable allows the executable to use the SHLIB_PATH to
#   look for its shared libraries.
#
$(TARGET): $(OBJECTS) Makefile
        $(LINK_EXE.f90) $(FFLAGS) -o $(TARGET) $(OBJECTS) $(LIBS); \
       chatr +s enable $(TARGET)
#
# Include Standard GLAS Dependencies
#
include $(UTILDIR)/make_depends.incl
#
# End of MakeFile
#
```

Abbreviations & Acronyms

ALT	designation for the EOS-Altimeter spacecraft series
DAAC	Distributed Active Archive Center
EDOS	EOS Data and Operations System
EOC	EOS Operating Center
EOS	NASA Earth Observing System Mission Program
EOSDIS	Earth Observing System Data and Information System
GDS	GLAS Ground Data System
GLAS	Geoscience Laser Altimeter System instrument or investigation
GPS	Global Positioning System
GSFC	NASA Goddard Space Flight Center at Greenbelt, Maryland
GSFC/WFF	NASA Goddard Space Flight Center/Wallops Flight Facility at Wallops Island, Virginia
ID	Identification
IEEE	Institute for Electronics and Electrical Engineering
IST	GLAS Instrument Support Terminal
LASER	Light Amplification by Stimulated Emission of Radiation
LIDAR	Light Detection and Ranging
N/A	Not (/) Applicable
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
POD	Precision Orbit Determination
SCF	GLAS investigation Science Computing Facility and workstation(s)
SDPS	Science Data Processing Segment
TBD	to be determined, to be done, or to be developed
UNIX	the operating system jointly developed by the AT&T Bell Laboratories and the University of California-Berkeley System Division

Glossary

aggregate	A collection, assemblage, or grouping of distinct data parts together to make a whole. It is generally used to indicate the grouping of GLAS data items, arrays, elements, and EOS parameters into a data record. For example, the collection of Level 1B EOS Data Parameters gathered to form a one-second Level 1B data record. It could be used to represent groupings of various GLAS data entities such as data items aggregated as an array, data items and arrays aggregated into a GLAS Data Element, GLAS Data Elements aggregated as an EOS Data Parameter, or EOS Data Parameters aggregated into a Data Product record.
array	An ordered arrangement of homogenous data items that may either be synchronous or asynchronous. An array of data items usually implies the ability to access individual data items or members of the array by an index. An array of GLAS data items might represent the three coordinates of a georeference location, a collection of values at a rate, or a collection of values describing an altimeter waveform.
file	A collection of data stored as records and terminated by a physical or logical end-of-file (EOF) marker. The term usually applies to the collection within a storage device or storage media such as a disk file or a tape file. Loosely employed it is used to indicate a collection of GLAS data records without a standard label. For the Level 1A Data Product, the file would constitute the collection of one-second Level 1A data records generated in the SDPS working storage for a single pass.
header	A text and/or binary label or information record, record set, or block, prefacing a data record, record set, or a file. A header usually contains identifying or descriptive information, and may sometimes be embedded within a record rather than attached as a prefix.
item	Specifically, a data item. A discrete, non-decomposable unit of data, usually a single word or value in a data record, or a single value from a data array. The representation of a single GLAS data value within a data array or a GLAS Data Element.
label	The text and/or binary information records, record set, block, header, or headers prefacing a data file or linked to a data file sufficient to form a labeled data product. A standard label may imply a standard data product. A label may consist of a single header as well as multiple headers and markers depending on the defining authority.
Level 0	The level designation applied to an EOS data product that consists of raw instrument data, recorded at the original resolution, in time order, with any duplicate or redundant data packets removed.
Level 1A	The level designation applied to an EOS data product that consists of reconstructed, unprocessed Level 0 instrument data, recorded at the full resolution with time referenced data records, in time order. The data are annotated with ancillary information including radiometric and geometric calibration coefficients, and georeferencing parameter data (i.e., ephemeris data). The included, computed coefficients and parameter data have not however been applied to correct the Level 0 instrument data contents.

Level 1B	The level designation applied to an EOS data product that consists of Level 1A data that have been radiometrically corrected, processed from raw data into sensor data units, and have been geolocated according to applied georeferencing data.
Level 2	The level designation applied to an EOS data product that consists of derived geophysical data values, recorded at the same resolution, time order, and georeference location as the Level 1A or Level 1B data.
Level 3	The level designation applied to an EOS data product that consists of geophysical data values derived from Level 1 or Level 2 data, recorded at a temporally or spatially resampled resolution.
Level 4	The level designation applied to an EOS data product that consists of data from modeled output or resultant analysis of lower level data that are not directly derived by the GLAS instrument and supplemental sensors.
metadata	The textual information supplied as supplemental, descriptive information to a data product. It may consist of fixed or variable length records of ASCII data describing files, records, parameters, elements, items, formats, etc., that may serve as catalog, data base, keyword/value, header, or label data. This data may be parsable and searchable by some tool or utility program.
orbit	The passage of time and spacecraft travel signifying a complete journey around a celestial or terrestrial body. For GLAS and the EOS ALT-L spacecraft each orbit starts at the time when the spacecraft is on the equator traveling toward the North Pole, continues through the equator crossing as the spacecraft ground track moves toward the South Pole, and terminates when the spacecraft has reached the equator moving northward from the South Polar region.
model	A graphical representation of a system.
module	A collection of program statements with four basic attributes: input and output, function, mechanics and internal data.
parameter	Specifically, an EOS Data Parameter. This is a defining, controlling, or constraining data unit associated with a EOS science community approved algorithm. It is identified by an EOS Parameter Number and Parameter Name. An EOS Data Parameter within the GLAS Data Product is composed of one or more GLAS Data Elements
pass	A sub-segment of an orbit, it may consist of the ascending or descending portion of an orbit (e.g., a descending pass would consist of the ground track segment beginning with the northernmost point of travel through the following southernmost point of travel), or the segment above or below the equator; for GLAS the pass is identified as either the northern or southern hemisphere portion of the ground track on any orbit
PDL	Program Design Language (Pseudocode). A language tool used for module programming and specification. It is at a higher level than any existing compilable language.
process	An activity on a dataflow diagram that transforms input data flow(s) into output data flow(s).

product	Specifically, the Data Product or the EOS Data Product. This is implicitly the labeled data product or the data product as produced by software on the SDPS or SCF. A GLAS data product refers to the data file or record collection either prefaced with a product label or standard formatted data label or linked to a product label or standard formatted data label file. Loosely used, it may indicate a single pass file aggregation, or the entire set of product files contained in a data repository.
program	The smallest set of computer instructions that can be executed as a stand-alone unit
record	A specific organization or aggregate of data items. It represents the collection of EOS Data Parameters within a given time interval, such as a one-second data record. It is the first level decomposition of a product file.
Scenario	A single execution path for a process.
Standard Data Product	Specifically, a GLAS Standard Data Product. It represents an EOS ALT-L/ GLAS Data Product produced on the EOSDIS SDPS for GLAS data product generation or within the GLAS Science Computing Facility using EOS science community approved algorithms. It is routinely produced and is intended to be archived in the EOSDIS data repository for EOS user community-wide access and retrieval.
State Transition Diagram	Graphical representation of one or more scenarios.
Stub	(alias dummy module) a primitive implementation of a subordinate module, which is normally used in the top-down testing of superordinate (higher) modules.
Structure Chart	A graphical tool for depicting the partitioning of a system into modules, the hierarchy and organization of those modules, and the communication interfaces between the modules.
Structured Design	The development of a blueprint of a computer system solution to a problem, having the same components and interrelationships amount the components as the original problem has.
Subroutine	A program that is called by another program
variable	Usually a reference in a computer program to a storage location, i.e., a place to contain or hold the value of a data item.